

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

ADOBE INC.,

Petitioner

v.

EXPRESS MOBILE, INC.,

Patent Owner

Case IPR2021-_____

U.S. Patent No. 9,063,755

DECLARATION OF IAN CULLIMORE, PH.D. IN SUPPORT OF

PETITION FOR *INTER PARTES* REVIEW OF

U.S. PATENT NO. 9,063,755

| |
|--|
| Adobe Inc. v. Express Mobile, Inc., IPR2021-XXXXX U.S. Pat. 9,063,755 Exhibit 1002 |
|--|

TABLE OF CONTENTS

| | | |
|--------------|--|-----|
| I. | INTRODUCTION | 1 |
| II. | BACKGROUND AND QUALIFICATIONS | 1 |
| III. | MATERIALS REVIEWED | 5 |
| IV. | LEVEL OF ORDINARY SKILL IN THE ART | 8 |
| V. | RELEVANT LEGAL STANDARDS | 9 |
| VI. | OVERVIEW OF THE '755 PATENT | 12 |
| VII. | CLAIM CONSTRUCTION | 19 |
| VIII. | STATE OF THE ART | 25 |
| | A. The Internet, Websites, and Web Browsers | 25 |
| | B. Web Services and Registries | 26 |
| IX. | OVERVIEW OF THE PRIOR ART | 31 |
| | A. <i>Huang</i> | 31 |
| | B. <i>Shenfield</i> | 47 |
| X. | THE PRIOR ART DISCLOSES OR SUGGESTS ALL OF THE FEATURES OF THE CHALLENGED CLAIMS | 50 |
| | A. Grounds Challenging Claims 1, 3, 5-7, 11-12, 14, 16-18, and 22 of the '755 patent | 50 |
| | B. Ground 1: <i>Huang</i> in View of <i>Shenfield</i> Render Claims 1, 3, 5-7, 11- 12, 14, 16-18, and 22 of the '755 Patent Obvious | 50 |
| | 1. Claim 1 | 51 |
| | 2. Claim 3 | 130 |

| | | |
|------------|------------------------|------------|
| 3. | Claim 5 | 134 |
| 4. | Claim 6 | 135 |
| 5. | Claim 7 | 137 |
| 6. | Claim 11 | 138 |
| 7. | Claim 12 | 141 |
| 8. | Claim 14 | 147 |
| 9. | Claim 16 | 147 |
| 10. | Claim 17 | 148 |
| 11. | Claim 18 | 148 |
| 12. | Claim 22 | 149 |
| XI. | CONCLUSION..... | 150 |

I, Ian Cullimore, declare as follows:

I. INTRODUCTION

1. I have been retained by Petitioner Adobe Inc. (“Adobe” or “Petitioner”) as an independent expert consultant in this proceeding before the United States Patent and Trademark Office (“PTO”).

2. I am being compensated at a rate of \$500/hour for my services in this proceeding, which is my regular and customary rate.

3. My compensation is in no way contingent on the nature of my findings, the presentation of my findings in testimony or this declaration, or the outcome of this or any other proceeding. I have no other interest in this proceeding.

4. I have been asked to consider whether certain references disclose and/or suggest the features recited in claims 1, 3, 5-7, 11-12, 14, 16-18, and 22 of U.S. Patent No. 9,063,755 (“the ’755 patent”) (Ex-1001).¹ My opinions are set forth below.

II. BACKGROUND AND QUALIFICATIONS

5. I am an independent consultant. All of my opinions stated in this declaration are based on my own personal knowledge and professional judgment. In forming my opinions, I have relied on my education, knowledge, and some 40 years

¹ Where appropriate, I refer to exhibits I understand will be attached to the petition for *inter partes* review of the ’755 patent (the “Petition”).

of experience in the field of computer science and software engineering, including, among other things, the development of interfaces for web services and programs designed to operate across multiple devices.

6. I received a bachelor's degree [B.Sc. (Hons)] in Mathematics from King's College, London, in 1981. Additionally, I received a Ph.D. in Cognitive and Computer Science from the University of Sussex in 2000. I also attended an M.Sc. course at Imperial College, London, in 1985. I was for a time an Honorary Senior Research Fellow at the University of Birmingham, U.K. I am currently a Visiting Lecturer at the University of Hereford, U.K. I have undertaken various other Continuous Professional Development courses over the years.

7. For the majority of my working career, I have been self-employed as a software, firmware, and hardware designer and developer, with a specialization in operating system design and development for mobile devices, ultra-low power management, IoT ("Internet of Things") devices, and computer communications. I remain a very "hands-on" developer for both software and hardware. I run my own consulting company, Crushproof Ltd., and from time to time am involved in other startups, as well as software and hardware, and business development, consulting for other companies. I currently reside in the U.K., but I have lived and worked extensively in the U.S., especially in Silicon Valley.

8. I have been involved in the design and development of mobile and wireless device software and hardware since 1984. From 2012 to 2015, I served as Chief Technology Officer at Paqet Systems Corp., a startup I co-founded that specializes in ultra-low power software operating systems for machine-to-machine, sensor hubs, wearables, Internet of Things, and other cutting-edge mobile technologies. Prior to that, from 2010 to 2012, I was Chief Technology Officer at IOTA Computing, Inc., another startup I co-founded that focused on software development for similar technology. Both companies were based in Silicon Valley. Between 2008 and 2010, I was an independent consultant and provided software and hardware development guidance for a series of technology companies. From 2006 to 2008, I worked as Chief Technology Officer at Martian Gaming Ltd., a London-based online mobile gaming platform that I co-founded. Additionally, I led the development of mobile server and client development in the course of serving as Chief Technology Officer for Cibenix Ltd., a Dublin-based (Ireland) mobile software developer, from 2004 to 2006. From 1997 to 2003, I served as Chief Executive Officer and Chief Technology Officer at Informal Software, a company I co-founded to implement an interface I had developed as part of my doctoral research.

9. Additionally, I have extensive professional experience in computer system design of many disciplines: software (from low-level embedded to high-level

applications and User Interfaces), hardware and communications, as well as server technologies. From 1996 to 1997, I worked as Chief Technology Officer and consultant to Two Way TV, an innovative interactive television company that developed user-interactive television and games. Between 1994 and 1996, I served as Software Director at Xircom, Inc., where I created award-winning Ethernet and modem installer software products and software for mobile Palm handheld devices to enable receipt and transmission of text messages, as well as Local Area Network connectivity for handhelds and laptops via innovative adapters and the first PC Card (“PCMCIA”) LAN adapters. Prior to that, from 1992 to 1994, I developed DOS power management systems and a secure-login software client at Digital Research Inc., acquired by Novell Inc. From 1988 to 1991, I was Vice President of Software Research and Development at Poqet Computer Corp., which I also co-founded; in that role, I developed power-managing ROM BIOS and created other software and hardware power management techniques to enhance battery life. Between 1986 and 1988, I served as Chief Technology Officer at D.I.P. Ltd., a company I co-founded that developed the world’s first “pocket PC.” From 1984 to 1986, I was a Software Engineer at Psion Ltd., a video game developer; there, I developed a PDA operating system and communications software. From 1983 to 1984, I was a Video Games Programmer at Thorn EMI, and from 1981 to 1983, I was a Software Programmer

at British Aerospace, where I wrote various programs for mainframes (including for the Concorde program).

10. I have served as a technical consultant in connection with intellectual property relating to various computer and mobile device engineering technologies since 1985. In just the past five years, I have been engaged as a consultant on ten projects, many of which involved mobile software and hardware development.

11. My work has involved projects specifically including web services and the development of software capable of operating on multiple devices, operating systems, and/or platforms. For example, I have developed an interface for web services for Toyota's salesforce designed to operate on hand-held devices to connect to back-end servers. I have also developed for a group at Sheffield University a device-dependent program to run device-independent XML content to dynamically present content to and obtain information from a user.

12. I am over 18 years of age and, if I am called upon to do so, I would be competent to testify as to the matters set forth herein. I have provided a copy of my curriculum vitae as Ex-1003, which contains additional details regarding my background and experience.

III. MATERIALS REVIEWED

13. The opinions contained in this declaration are based on the documents I reviewed, my professional judgment, as well as my education, experience, and

knowledge regarding inventions, electrical engineering, computer science, web services, registries, and efforts in the field to achieve cross-functionality on various platforms, such as mobile phones.

14. In forming my opinions expressed in this declaration, I reviewed the following materials:

- The '755 patent (Ex-1001);
- Prosecution History of the '755 patent (Ex-1004);
- U.S. Patent Application Publication No. 2007/0118844 to Jin Huang et al. ("*Huang*") (Ex-1005);
- U.S. Patent Application No. 2007/0079282 to Pawan Nachnani et al. ("*Nachnani*") (Ex-1006);
- U.S. Patent Application No. 2006/0200749 to Michael Shenfield ("*Shenfield*") (Ex-1007);
- Memorandum Opinion (Dkt. 121), *Express Mobile, Inc. v. GoDaddy.com, LLC*, C.A. No. 19-1937-RGA (D. Del. June 1, 2021) (Ex-1011);
- Memorandum Order (Dkt. 137), *Shopify Inc. et al. v. Express Mobile, Inc.*, C.A. No. 19-439-RGA (D. Del. June 23, 2020) (Ex-1012);

- Joint Claim Construction Brief (Dkt. 117), *Shopify Inc. et al. v. Express Mobile, Inc.*, C.A. No. 19-439-RGA (lead case) (D. Del. May 8, 2020) (Ex-1015);
- Alan Grosskurth and Michael W. Godfrey, A Reference Architecture for Web Browsers, *Journal of Software Maintenance & Evolution: Research & Practices* 2006; 00:1-7 (2006) (Ex-1016);
- Hypertext Markup Language - 2.0, IETF (Nov. 1997), *available at* <https://datatracker.ietf.org/doc/html/rfc1866> (Ex-1017);
- Francisco Curbera et al., Unraveling the Web Services Web, *IEEE Internet Computing* Vol. 7, Issue 2 86-93 (Apr. 2002) (Ex-1018);
- U.S. Patent Application Publication No. 2007/0067421 (Ex-1019);
- Google Search WSDL Document, *available at* <https://cs.au.dk/~amoeller/WWW/webservices/GoogleSearch.wsdl> (Rev. Aug. 16, 2002) (Ex-1020);
- U.S. Patent Application Publication No. 2003/0009523 (Ex-1021);
- U.S. Patent Application Publication No. 2006/0034202 (Ex-1022);
- U.S. Patent Application Publication No. 2004/0186888 (Ex-1023);
- U.S. Patent Application Publication No. 2006/0221190 (Ex-1024);

- U.S. Patent Application Publication No. 2007/0100836 (Ex-1025);
- U.S. Patent No. 7,191,160 (Ex-1026);
- Steven J. Vaughan-Nichols, Web Services: Beyond the Hype, Computer 35(2):18-21 (Feb. 2002) (Ex-1027);
- U.S. Patent No. 7,873,625 (Ex-1028); and
- U.S. Patent Application Publication No. 2006/0230462 (Ex-1029).

IV. LEVEL OF ORDINARY SKILL IN THE ART

15. I was asked to provide my opinion on the level of one of ordinary skill in the art with respect to the alleged invention of the '755 patent as of April 7, 2008. Based on my review of the types of problems encountered in the art, prior solutions to those problems, the rapidity with which innovations were made, the sophistication of the technology, and the educational level of active workers in the field, I believe a person of ordinary skill in the art would have had a Bachelor of Science degree in Computer Science, Computer Engineering, or equivalent, plus two years working experience. More education can supplement practical experience and vice versa.

16. All of my opinions in this declaration are from the perspective of one of ordinary skill in the art as I have defined it here during the relevant timeframe (i.e., mid-2008) (“POSITA”).

V. RELEVANT LEGAL STANDARDS

17. I am not an attorney and offer no legal opinions, but in the course of my work, I have had experience studying and analyzing patents and patent claims from the perspective of a POSITA.

18. For the purposes of this declaration, I have been informed about certain aspects of the law that are relevant to forming my opinions. My understanding of the law is as follows:

19. Petitioner's counsel has informed me that a patent claim is invalid as "anticipated" if each and every limitation of the claim is found, either expressly or inherently, in a single device or method that predates the claimed invention or is described in a single publication or patent that predates the claimed invention. Petitioner's counsel has informed me that, in patent law, these previous devices, methods, publications, or patents are called "prior art."

20. Petitioner's counsel has informed me that claim limitations that are not expressly found in a prior art reference are inherent if the prior art necessarily functions in accordance with, or includes, the claim limitations. I understand that, for a claimed limitation to be inherently present, the prior art need not expressly disclose the limitation, so long as the claimed limitation necessarily flows from a disclosure in the prior art. I also understand that it is acceptable to examine evidence outside the prior art reference (extrinsic evidence) in determining whether a feature,

while not expressly discussed in the reference, is necessarily present in that reference.

21. Petitioner's counsel has informed me that a patent claim can be considered to have been obvious to a POSITA at the time the application was filed. I am informed this means that, even if all of the requirements of a claim are not found in a single prior art reference, the claim is not patentable if the differences between the subject matter in the prior art and the subject matter in the claim would have been obvious to a POSITA at the time the of the invention.

22. Petitioner's counsel has informed me that a determination of whether a claim would have been obvious should be based upon several factors, including, among others:

- the level of ordinary skill in the art at the time the application was filed;
- the scope and content of the prior art; and
- what differences, if any, existed between the claimed invention and the prior art.

23. Petitioner's counsel has informed me that a single reference can render a patent claim obvious if any differences between that reference and the claims would have been obvious to a POSITA. Alternatively, I understand that the teachings of two or more references may be combined in the same way as disclosed in the claims, if such a combination would have been obvious to one having ordinary

skill in the art. In determining whether a combination based on either a single reference or multiple references would have been obvious, it is appropriate to consider, among other factors:

- whether the teachings of the prior art references disclose known concepts combined in familiar ways, and when combined, would yield predictable results;
- whether a POSITA could implement a predictable variation, and would see the benefit of doing so;
- whether the claimed elements represent one of a limited number of known design choices, and would have a reasonable expectation of success by those skilled in the art;
- whether a person of ordinary skill would have recognized a reason to combine known elements in the manner described in the claim;
- whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent; and
- whether the innovation applies a known technique that had been used to improve a similar device or method in a similar way.

24. Petitioner's counsel has informed me that a POSITA has ordinary creativity and is not an automaton. Petitioner's counsel has also informed me that,

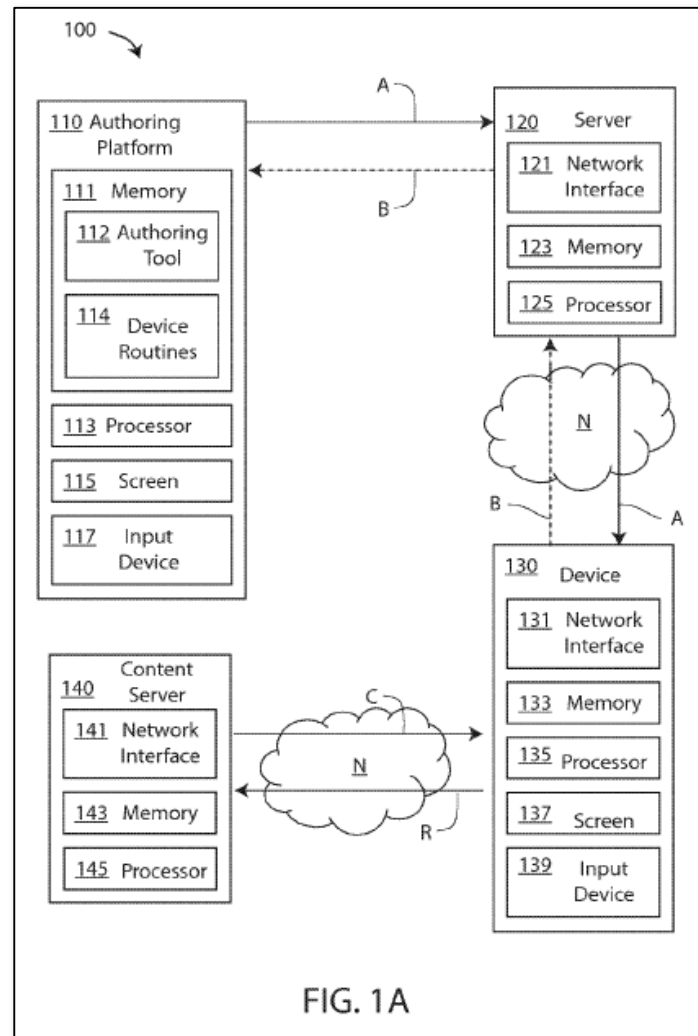
in considering obviousness, obviousness may not be determined using the benefit of hindsight, including hindsight derived from the patent being considered.

VI. OVERVIEW OF THE '755 PATENT

25. The '755 patent is titled “Systems and Methods for Presenting Information on Mobile Devices.” (Ex-1001, Cover.) It “generally relates to providing software for mobile devices, and more particularly to a method and system for authoring Applications for devices.” (*Id.*, 1:8-10; *see also* Ex-1004, 1209-16 (applicant summary of '755 patent during prosecution).)

26. The '755 patent describes a “system” with “a database of web services obtainable over a network and an authoring tool” that “generate[s] code to provide content on a display of a platform.” (Ex-1001, 1:34–37.) The '755 patent states that “[t]he authoring tool is configured to define an object for presentation on the display, select a component of a web service included in said database, associate said object with said selected component, and produce code that, when executed on the platform, provides said selected component on the display of the platform.” (*Id.*, 1:37-42.) One “mechanism for binding the outputs of the web service to the UI components is through symbolic references that matches each output to the symbolic name of the UI component.” (*Id.*, 9:11-14.) The generated code includes a device-independent “Application” and a device-dependent “Player.” (*See id.*, Abstract, 1:59–2:3.)

27. The '755 patent explains that Figure 1A, reproduced below, shows “a system including an authoring platform and a server for providing programming instructions to a device over a network.” (Ex-1001, 2:14-17.)



(Ex-1001, Fig. 1A.)

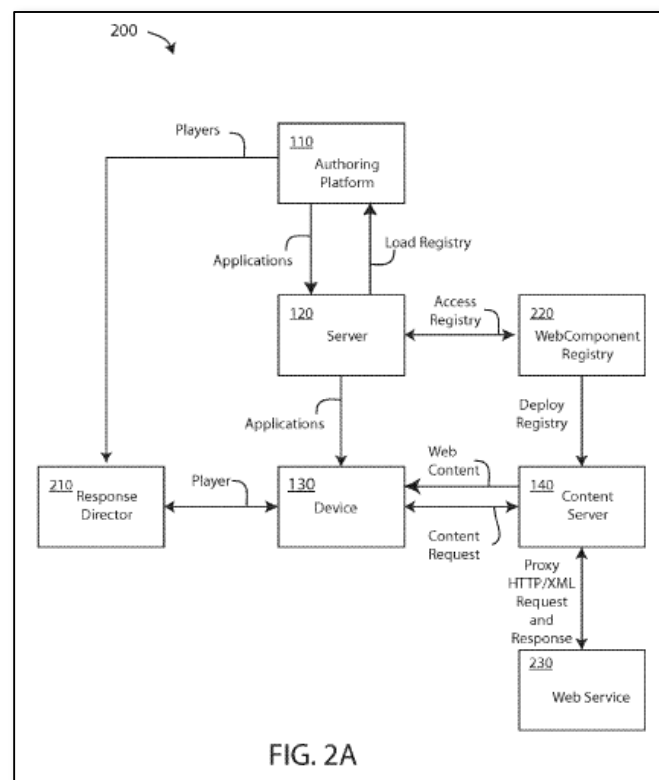
Specifically, Figure 1A “is an illustrative schematic” of “a system 100 including an authoring platform 110 and a server 120 for providing programming instructions to a device 130 over a network N.” (*Id.*, 2:65-3:3.) The '755 patent explains that “[i]n

general, a user of authoring platform 110 may produce programming instructions or files that may be transmitted over network N to operate device 130, including instructions or files that are sent to device 130 and/or server 120.” (*Id.*, 3:5-9.)

28. The ’755 patent explains that “[i]n one embodiment, system 100 provides permits [*sic*] a user of authoring platform 110 to provide instructions to each of the plurality of devices 130 in the form of a device- or device-platform specific instructions for processor 135 of the device, referred to herein and without limitation as a ‘Player,’ and a device-independent program, referred to herein and without limitation as an ‘Application.’” (Ex-1001, 5:8-15.) “[D]evice 130 utilizes a Player and an Application to execute programming from authoring platform 110” and, because “[t]he programming is parsed into instructions used by different device platforms and instructions that are independent of device platform,” “authoring platform 110 may be used to generate programming for a plurality of devices 130 having one of several different device platforms.” (*Id.*, 5:15-24.) Put differently, one device-independent Application can run on multiple devices through the use of multiple device-specific Players that translate the Application into instructions that can be recognized by the specific device. (*See id.*, 5:42-55, 6:4-17.) The ’755 patent contemplates that the “Player” need not necessarily be a stand-alone executable, and it can be “activated by a web browser or other software on device 130.” (Ex-1001, 5:27-28.)

29. The '755 patent explains that “code in a device-independent format,” which the '755 patent refers to as a “Portable Description Language (PDL),” is “conceptionally viewed as a device, operating system and virtual machine agnostic representation of Java serialized objects.” (Ex-1001, 6:4-6, 6:49-51.) As a result, a device-independent Application coded in PDL “no longer ha[s] any dependencies from the original programming language (Java) that they were defined in.” (*Id.*, 7:14-17.)

30. Figure 2A of the '755 patent, reproduced below, “illustrat[es] the communications between different system components.” (*Id.*, 7:63-65.)



(Ex-1001, Fig. 2A.)

31. The '755 patent explains that “authoring platform 110 permits a user to design desired displays for screen 137 and actions of device 130”—i.e., “authoring platform 110 is used to program the operation of device 130.” (Ex-1001, 4:13-16.) One aspect of the authoring system is that it allows “a user to assign actions to one or more of an input object, an output object, or an action object,” and the user can “bind one or more of [such objects] with web services or web components.” (*Id.*, 6:37-47.) The '755 patent explains that “[w]eb service 230 is a plurality of services obtainable over the Internet.” (*Id.*, 8:18-19.)

32. The '755 patent explains that “[a] user of authoring platform 110 of system 200 may define associations with web services as WebComponent Bindings,” such as “associat[ing] certain objects for display that provide input or output to components of web service 230.” (Ex-1001, 8:48-52.) The '755 patent discloses the use of “web component registry 220,” which “may contain consumer inputs related to each web service 230, environmental data such as PIM, time or location values, persistent variable data, outputs related to the web service, and/or optional hinting for improving the user’s productivity,” “so that a user of the authoring platform may bind web services 230 to elements to be displayed on device 130. (*Id.*, 8:22-26, 8:36-47.) In one embodiment of the '755 patent, “the mechanism for binding the outputs of the web service to the UI components is through symbolic

references that match each output to the symbolic name of the UI component.” (*Id.*, 9:11-14.)

33. At a high level, the ’755 patent relates to systems and methods that allow for designing cross-functional web pages, utilizing device-independent Applications that define the content and a set of device-specific Players that translate the Applications to run on various devices. The Applications contain user interface objects that may be bound to web services through “symbolic references” or “symbolic names.”

34. Exemplary claims 1 and 12 of the ’755 patent are representative of the independent claims of the ’755 patent. They are reproduced below.

1. A system for generating code to provide content on a display of a device, said system comprising:
 - computer memory storing a registry of:
 - a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and
 - b) the address of the web service;
 - an authoring tool configured to:
 - define a user interface (UI) object for presentation on the display, where said UI object corresponds to the web component included in said registry selected from the group consisting of an input of the web service and an output of the web service,
 - access said computer memory to select the symbolic name corresponding to the web component of the defined UI object,

associate the selected symbolic name with the defined UI object,
 produce an Application including the selected symbolic name of the defined UI object, where said Application is a device-independent code, and
 produce a Player, where said Player is a device-dependent code;
 such that, when the Application and Player are provided to the device and executed on the device, and when a user of the device provides one or more input values associated with an input symbolic name to an input of defined UI object,

- 1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service,
- 2) the web service utilizes the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name,
- 3) said Player receives the output symbolic name and corresponding one or more output values and provides instructions for a display of the device to present an output value in the defined UI object.

12. A method of displaying content on a display of a device utilizing a registry of one or more web components related to inputs and outputs of a web service obtainable over a network, where each web component includes a plurality of symbolic names of inputs and outputs associated with each web service, and where the registry includes: a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and

b) the address of the web service, said method comprising:
 defining a user interface (UI) object for presentation on the display, where said UI object corresponds

to a web component included in said registry
selected from the group consisting of an input of
the web service and an output of the web service;
selecting a symbolic name from said web
component corresponding to the defined UI
object;
associating the selected symbolic name with the
defined UI object;
producing an Application including the selected
symbolic name of the defined UI object, where
said Application is a device-dependent code; and
producing a Player, where said Player is a device-
dependent code;
such that, when the Application and Player are
provided to the device and executed on the
device, and when a user of the device provides
one or more input values associated with an
input symbolic name to an input of defined UI
object,
1) the device provides the user provided one or more
input values and corresponding input symbolic
name to the web service,
2) the web service utilizes the input symbolic name
and the user provided one or more input values
for generating one or more output values having
an associated output symbolic name,
3) said Player receives the output symbolic name
and corresponding one or more output values
and provides instructions for a display of the
device to present an output value in the defined
UI object.

VII. CLAIM CONSTRUCTION

35. I understand that claim terms are typically given their ordinary and customary meanings, as would have been understood by a POSITA at the time of the alleged invention. I have been asked to assume for purposes of this declaration

that the time of the alleged invention is the filing date of the earliest provisional application to which the '755 patent claims priority, which is April 7, 2008. I also understand that the meanings of claim terms are not determined in a vacuum but rather in the context of the claims, the specification, and the prosecution history.

36. I also understand that the Board may correct an obvious error in a claim if the correction is not subject to reasonable debate after considering the language of the claims and the patent specification and the prosecution history does not suggest a different interpretation of the claims.

37. I understand that Express Mobile has stipulated to the following claim constructions, which district courts have adopted:

| Term | Stipulated Construction |
|--|--|
| “where said Application is a device-dependent code” | “where said Application is a device-independent code” (Ex-1011, 10.) |
| “authoring tool” / “authoring tool configured to” | “a system, with a graphical interface, for generating code to display content on a device screen” (Ex-1011, 10.) |

| Term | Stipulated Construction |
|--|---|
| “device-dependent code” | “code that is specific to the operating system, programming language, or platform of a device” (Ex-1011, 10.) |
| “device-independent code” | “code that is not specific to the operating system, programming language, or platform of a device” (Ex-1011, 10.) |
| “for evoking one or more web components” | “for calling up one or more web components” (Ex-1011, 10.) |

38. I understand that district courts have provided claim constructions for the following terms:

| Term | Courts’ Construction | Exemplary Intrinsic Evidence |
|-----------------|--|---|
| “registry” | “a database that is used for computing functionality” (Ex-1011, 17-19.) | Ex-1001, 1:34-42, 1:51-58, 7:1-3, 7:63-66, 8:19-22, 22:26-29. |
| “web component” | “software objects that provide functionalities of a web service” (Ex-1011, 19-20.) | Ex-1001, 8:18-22, 8:40-42, Fig. 3F. |

| Term | Courts' Construction | Exemplary Intrinsic Evidence |
|-------------------------------|--|---|
| “web service” | “a software system that supports interaction between devices over a network” (Ex-1011, 20-22.) | Ex-1001, 8:18-19; Ex-1004, 1068. |
| “Application” / “application” | “device-independent code containing instructions for a device and which is separate from the Player” Ex-1011, 23-25.) | Ex-1001, 1:59-2:3, 5:8-15; Ex-1004, 214, 1216. |
| “Player” / “player” | “device-specific code which contains instructions for a device and which is separate from the Application” (Ex-1012, 9-14) | Ex-1001, 1:59-2:3, 5:8-15; Ex-1004, 1214, 1216. |

39. I have been asked to determine whether the district courts’ constructions are consistent with the intrinsic record. In my opinion, the district courts’ constructions are consistent with the disclosures of the specification and prosecution history. I have provided exemplary citations in the table above.

40. I have been asked to determine if there are any particular claim terms that should be construed for the Board to understand how the prior art references read on the challenged claims. Other than to correct an obvious error in the phrase “where said application is a device-dependent code” as used in claim 12, I have not identified any such claim terms.

41. With regard to claim 12, it is my opinion that a POSITA would have understood that the phrase “where said application is a device-dependent code” as used in claim 12 of the ’755 patent is an obvious error that should be read to mean “where said Application is a device-*in*dependent code.”²

42. First, it is my opinion that the correction is not subject to reasonable debate in view of the claim language and prosecution history. For example, the specification consistently describes the “Application” as “device-independent (Ex-1001, Abstract (“Applications that are device independent”), 2:1-2 (“an Application that is a device independent code”), 5:13-15 (“a device-independent program, referred to herein and without limitation as an ‘Application’”), 5:57-59 (“the Player transforms device-independent instructions of the Application into device-specific instructions that are executable by device 130”), 6:4-5 (“The Application is preferably code in a device-independent format”), 6:13-14 (“device-independent Applications”), 13:49 (“device-independent Application”).

² All emphases and color annotations added unless noted otherwise.

43. Further supporting my opinion is that I understand that Express Mobile has agreed to this correction in multiple district court litigations. (Ex-1011, 10; Ex-1015, 2.)

44. Further, I am not aware of any evidence in the prosecution history that would suggest a different interpretation of this phrase to a POSITA. Indeed, the prosecution history confirms that “where said Application is a device-dependent code” should mean “where said Application is a device-*in*dependent code.” For example, the applicant repeatedly asserting during prosecution that the claimed “Application” is device-independent code (*e.g.*, Ex-1004, 1017 (“an **Application** ... that is device-independent”), 1067 (same), 1150 (same), 1214 (same) (emphases in original)).

45. In my analysis below, I have applied the stipulated or district courts’ constructions of the terms in the table above. Otherwise, I have given all other claim terms of the challenged claims their plain and ordinary meaning, as would have been understood by a POSITA, at the time of the alleged invention, which I assume is April 2008.

46. I note, however, that in my opinion, my conclusions would not change even if I applied the plain meaning of the claim terms because the plain meaning would be consistent with or subsume the constructions of the district courts.

VIII. STATE OF THE ART

47. In this section, I discuss the state of the art with respect to technologies relevant to the subject matter of the '755 patent. Many of the technologies relevant to the '755 patent were well known and within the knowledge of a POSITA.

A. The Internet, Websites, and Web Browsers

48. A person of ordinary skill would have known in the relevant time frame about computer networks, including the Internet, websites on the Internet, and web browsers.

49. The Internet is a network of computers. The World Wide Web is a “universal information space operating on top of the Internet.” (Ex-1016, 2.) Resources on the Web are “identified by a unique Uniform Resource Identifier (URI).” (*Id.*) Resources include, among other things, “documents, images, sound clips, or video clips.” (*Id.*)

50. “Web pages are documents typically in HyperText Markup Language (HTML) format that the web browser retrieves from a web server computer via a network.” (Ex-1006, ¶[0006].) HTML documents are platform independent. (Ex-1017, 1.) Put differently, any device can render HTML web pages with the appropriate software, such as a web browser.

51. Web browsers “allows a user to view and interact with web pages.” (*E.g.*, Ex-1006, ¶[0006].) Specifically, web browsers “retrieve[] documents from

remote servers and display[] them on screen,” requesting resources utilizing the URI. (Ex-1016, 2.) Web browsers are generally installed directly on a device, such as a computer, phone, or other web-accessible device; therefore, web browsers are generally understood to be device-specific programs that visually display web pages that may be written, for example, in device-independent HTTP. (See Ex-1006, ¶[0006].)

52. As of April 2008, a POSITA would have been well aware of the Internet, the World Wide Web, Websites, and Web Browsers. Such technology was well established by this date. For example, Version 2.0 of the HTML standard had been established by November 1995 (Ex-1017, 1) and publications report numerous web browsers had been released by April 2008 (*e.g.*, Ex-1016, 4 (Fig. 1)).

B. Web Services and Registries

53. As of April 2008, a POSITA also would have been familiar with web services. Web services allow users to access processes through a network-based interface. Web services are Internet-based as early as April 2002, publications reported that “Web services are emerging to provide a systematic and extensible framework for application-to-application interaction, built on top of existing Web protocols and based on open XML standards.” (Ex-1018, 86.) By April 2008, web services had become a well-known way of achieving interoperability between different computer systems. (Ex-1005, ¶[0003].) As noted in the prior art, web

services were well known in the field and utilize open protocols and standards, such as XML, to exchange data with calling clients. (*Id.*, ¶¶[0003], [0005].)

54. Generally, three pieces of information are needed to access a web service: (1) what communication protocol to use, (2) how to accomplish individual service interactions over the protocol, and (3) where to terminate communication. (Ex-1018, 89.) Thus, to access a web service, a user would need to know the structure of messages for sending and receiving information, what information to send or will be received and the vocabulary to use in the message, and where to send the properly formatted message.

55. Because numerous web services were available and developed independently, web service registries and a uniform language to describe web services (the Web Services Description Language or WSDL) had been developed.

56. One well-known web service registry was the Uniform Discovery Description and Integration (UDDI) services registry. The UDDI registry is described as a “centralized registry of services that is roughly equivalent to an automated online ‘phone directory’ of Web services.” (Ex-1018, 90.) Put differently, the UDDI registry allowed users to locate web services without needing to separately reach out to each web service vendor. The UDDI registry lists what web services are available, the nature of the web service, and technical information,

such as how to encode and provide data to the web service—i.e., how to interface with the web service. (*Id.*, 90-91.)

57. Among the information typically found in a UDDI services registry is a WSDL description of the web service. (*See, e.g.*, Ex-1007, ¶¶[0025]; Ex-1019, ¶¶[0006]-[0007]; Ex-1018, 91 (discussing registering a WSDL document to UDDI registry).) A WSDL description of a web service provides an “abstract description [of a web service] in terms of messages exchanged in a service interaction.” (Ex-1018, 88.) Such information includes definitions of data types, the form of the message, the content of a message, and where to send the message. (*E.g.*, Ex-1019, ¶¶[0009]-[0015] & Fig. 2.)

58. For example, below are excerpts of a WSDL description for Google Search, which is available at <https://cs.au.dk/~amoeller/WWW/webservices/GoogleSearch.wsdl>. (Ex-1020.) The GoogleSearch WSDL defines the data types used in its web service interface. (*Id.*) The GoogleSearch WSDL also defines the structure and content of its messages, as shown in the excerpt below:

```

<!-- Messages for Google Web APIs - cached page, search, spelling. -->
▼<message name="doGetCachedPage">
  <part name="key" type="xsd:string"/>
  <part name="url" type="xsd:string"/>
</message>
▼<message name="doGetCachedPageResponse">
  <part name="return" type="xsd:base64Binary"/>
</message>
▼<message name="doSpellingSuggestion">
  <part name="key" type="xsd:string"/>
  <part name="phrase" type="xsd:string"/>
</message>
▼<message name="doSpellingSuggestionResponse">
  <part name="return" type="xsd:string"/>
</message>
<!-- note, ie and oe are ignored by server; all traffic is UTF-8. -->
▼<message name="doGoogleSearch">
  <part name="key" type="xsd:string"/>
  <part name="q" type="xsd:string"/>
  <part name="start" type="xsd:int"/>
  <part name="maxResults" type="xsd:int"/>
  <part name="filter" type="xsd:boolean"/>
  <part name="restrict" type="xsd:string"/>
  <part name="safeSearch" type="xsd:boolean"/>
  <part name="lr" type="xsd:string"/>
  <part name="ie" type="xsd:string"/>
  <part name="oe" type="xsd:string"/>
</message>
▼<message name="doGoogleSearchResponse">
  <part name="return" type="typens:GoogleSearchResult"/>
</message>

```

(Ex-1020, 2.)

59. As can be seen in this excerpt from the GoogleSearch WSDL example, each message is identified by name and has associated parts with identifiers and specifying the type of data to be included. For example, the message “doGetCachedPage” has a part named “key” that has data in the form of a string (i.e., a sequence of characters) and “URL,” which also has string data. (Ex-1020, 2.) For another example, the message “doGoogleSearch” has data identified as, for example, “start” and “maxResults.” (*Id.*)

60. Some data types are complex data types that are defined elsewhere in the WSDL document. For example, the message “doGoogleSearchResponse” has a single part named “return” that contains data of the type “GoogleSearchResult.”

(Ex-1020, 2.) The WSDL document identifies this data type as having a set of different types of data:

```
<xsd:complexType name="GoogleSearchResult">
  <xsd:all>
    <xsd:element name="documentFiltering" type="xsd:boolean"/>
    <xsd:element name="searchComments" type="xsd:string"/>
    <xsd:element name="estimatedTotalResultsCount" type="xsd:int"/>
    <xsd:element name="estimateIsExact" type="xsd:boolean"/>
    <xsd:element name="resultElements" type="typens:ResultElementArray"/>
    <xsd:element name="searchQuery" type="xsd:string"/>
    <xsd:element name="startIndex" type="xsd:int"/>
    <xsd:element name="endIndex" type="xsd:int"/>
    <xsd:element name="searchTips" type="xsd:string"/>
    <xsd:element name="directoryCategories" type="typens:DirectoryCategoryArray"/>
    <xsd:element name="searchTime" type="xsd:double"/>
  </xsd:all>
</xsd:complexType>
```

(Ex-1020, 1.)

Thus, the data type “GoogleSearchResult” has a number of elements of varying data types, including “documentFiltering,” which is a Boolean data type (i.e., true or false), “estimatedTotalResultsCount,” which is an integer (i.e., a whole number), and two additional complex data types (“resultElements” and “directoryCategories”), which are defined under the “types” section in the WSDL document. (Ex-1020, 1-2)

61. As of April 2008, a POSITA would also have been well aware of Web service registries and WSDL descriptions that could be found in Web service registries.

62. Well-known web services as of April 2008 include text communication or chat services (*e.g.*, Ex-1021, ¶[0002] (“Chat services on the Internet provide for real time communication between two users via a computer, wireless device, or any

other text based communication apparatus.... Most networks and online services offer some type of chat feature.”); Ex-1022, ¶[0004] (“Web service interface (WSI) messaging”); Ex-1023, ¶[0005] (“Another popular feature of the Internet are so-called ‘chat rooms’. Essentially, a chat room is a computer site that can be accessed (i.e., ‘logged onto’) simultaneously by many users, with each user being able to input text material intended to be conversational in nature.”)), image services (*e.g.*, Ex-1024, ¶[0006] (“a server hosting a media storage and display service”)), and RSS feeds (*e.g.*, Ex-1025, ¶[0003] (“Generally, RSS provides web content or summaries of web content together with links to the full versions of the content, and other meta data.”)).

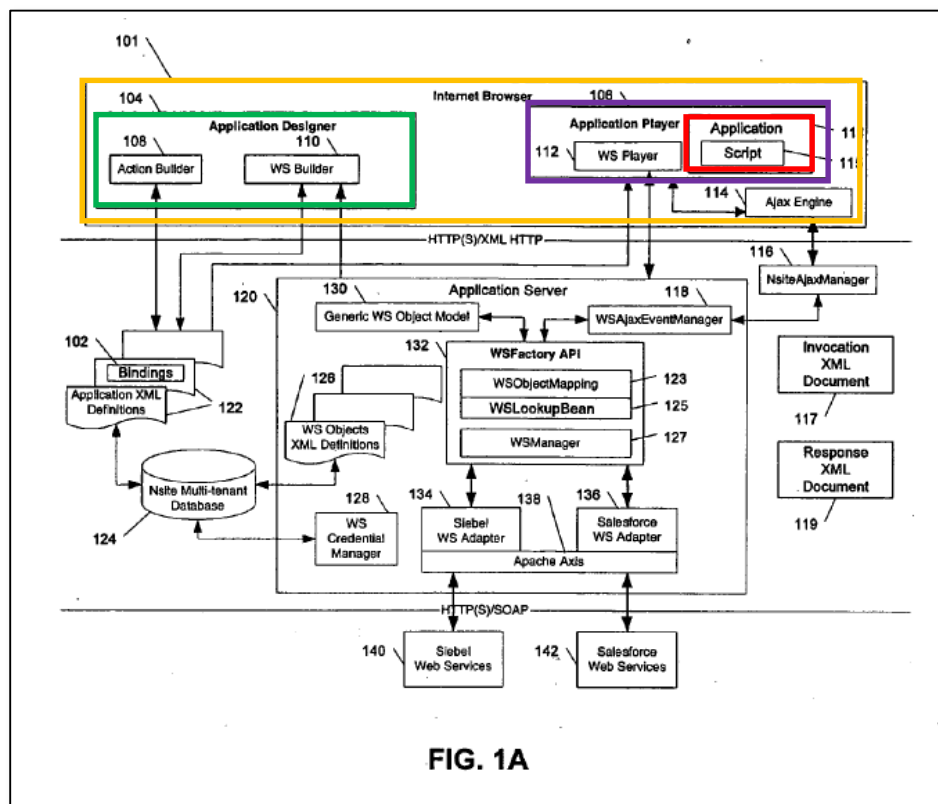
IX. OVERVIEW OF THE PRIOR ART

A. *Huang*

63. U.S. Patent Application No. 2007/0118844 to Jin Huang et al. (“*Huang*”) is titled “Designer and Player for Web Services Applications.” (Ex-1005, Cover (54).) *Huang* relates to a system that generates browser-based applications that execute in web browsers that utilize web services to allow for interoperability between systems having different operating systems. (*Id.*, ¶¶[0003], [0007].)

64. *Huang* discloses a “computer enabled system for developing and executing browser-based software applications.” (Ex-1005, ¶[0040].) *Huang*’s system includes a **browser 101** running a **designer 104**, which generates an

application 111 that will be executed on **player 106**, which also executes within **browser 101**, and where the **browser** communicates with an application **server 120**³:



(Ex-1005, Fig. 1A; *see also id.*, ¶¶[0040]–[0043].)

65. *Huang* discloses providing a player to an internet browser and providing an application that is loaded into the player for execution. (Ex-1005, ¶¶[0040]–[0042].) Although *Huang*’s disclosures focus on a browser-based embodiment, *Huang* explains that its system is “not so limited” to web browsers.

³ Where appropriate, I have color coded text and any corresponding aspects of figures in the same color. Otherwise, I have annotated figures using yellow highlighting.

(*Id.*, ¶[0008].) A POSITA would have understood that alternatives to browser-based embodiments would include native applications designed to run within the runtime environment of a device.

66. For the browser-based embodiment, *Huang* explains that the player “execute[s] in a conventional internet browser” and “provides a user interface which executes a browser based application.” (*Id.*, ¶[0040].) More specifically, the designer creates applications that “are stored in Extensible Markup Language (XML) format as XML definitions 122 in a database 124, and loaded into Player 106 as the application 111.” (*Id.*, ¶[0042].)

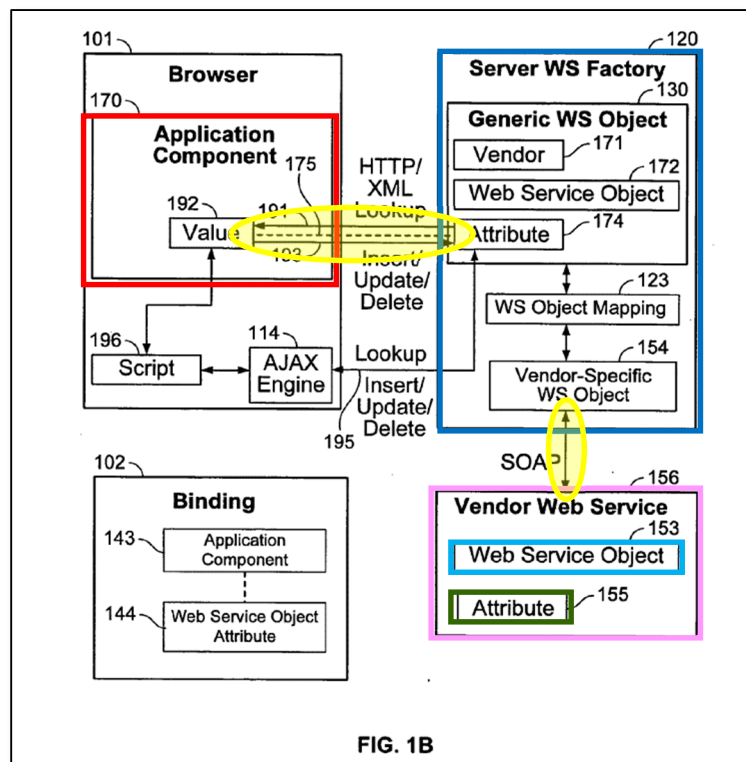
67. It is my opinion that a POSITA would have understood that XML is device-independent code. In particular, a POSITA as of April 2008 would have understood that XML is a data structure that uses markup language with tags to specify the content and structure of the data and that XML is designed to operate across platforms and devices. (Ex-1026, 4:3-8 (“*[d]ata that complies with the XML Standard is platform-independent* because the data’s content and structure is specified by the tags”); *see also id.*, 3:64-4:3 (“[D]ata types 216 are *platform-independent because they comply with the XML Standard.*”).)

68. *Huang* also explains that its Player includes a “WS Player 112 component,” which “allows applications executing in the Player 106 to interact with web services by, for example, looking up a value from a web service and storing the

value in an application component associated with the application 111, or inserting, updating, or deleting a value in a web service object, where the value is specified by an application component.” (*Id.*, ¶[0054].) Thus, the Player allows the XML applications to interact with web services.

69. *Huang* explains that the application has “application components” that are bound to web service object attributes. (Ex-1005, ¶[0042].) A POSITA would have understood that *Huang* describes “application components” as visual elements that may serve as user interfaces, both to input data and to display data. (*E.g.*, Ex-1005, ¶[0057] (“The application component includes a value 192, which is displayed to a user and may be set by a user, e.g., as a value of a text box or of a drop-down list.”).) These application components are “bound” to web service attributes—in Figure 1B, application component 170 is bound to attribute 155 of web service 156. (*Id.*, ¶[0056].) Such “bindings,” such as binding 193, are “a two-way mapping which specifies that a particular web service attribute 155 can be set to the value 192 of the application component 170, and, conversely, that the value of the application component 170 can be set to the value of the attribute 155.” (*Id.*) A POSITA would have understood that these web service attributes correspond to the inputs and outputs of a web service in view of *Huang*’s disclosure that these bindings allow the application to “set and get values of the attribute 155 of the web service.” (*Id.*, ¶¶[0047], [0058], [0106]-[0107]; Figs. 4A-4C.)

70. Figure 1B of *Huang* shows the relationship between the elements of *Huang*'s system when interacting with a web service. As can be seen in the figure below, **application component 170** can get or send a data value (shown in the two-way arrow) via **server 120** to **web service 156**. More specifically, **application component 170** is bound to **attribute 155** of **web service object 153** and can send an input to **web service 156** (set the value of **attribute 155** to the value of the **application component 170**) or get an output from **web service 156** (get the value of **attribute 155** and insert it into the value of **application component 170**):



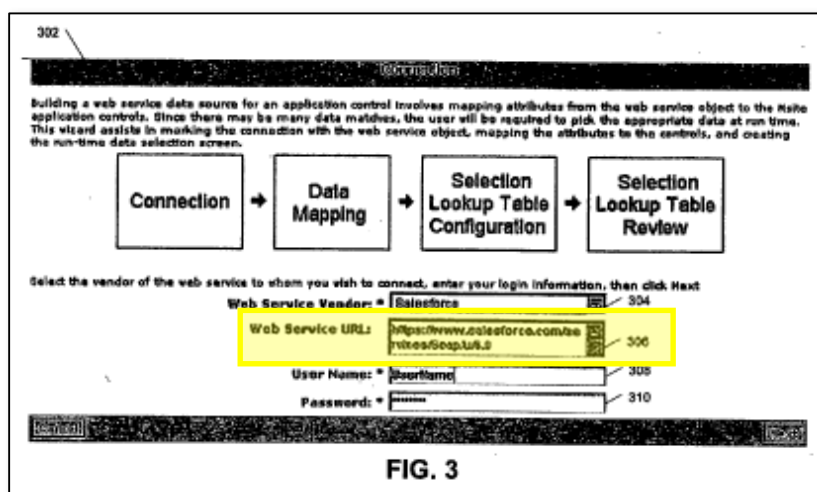
(Ex-1005, Fig. 1B.)

71. *Huang* implements an abstraction layer that allows applications to directly interact with generic objects that are mapped to vendor-specific inputs and outputs. (Ex-1005, ¶[0074].) By taking this approach, an application is not tied to any particular web services vendor. (*Id.*; *see also id.*, ¶[0057] (“To allow the Designer 104 and Player 106 to be vendor-independent and work with multiple web services without the need for vendor-specific code in the Designer 104, the Player 106, or the Internet Browser 101, a Generic WS Object 130 is provided in the server 120.”).) *Huang* explains that this enables applications to “build composite software applications which draw objects from several different organizations.” (*Id.*, ¶[0008].) *Huang* further discloses “adapters” that relate the generic objects to the vendor-specific objects needed to access a web service. (*Id.*, ¶[0044].) *Huang* explains that these relationships are stored as an XML definition or an XML mapping file and is generated from, for example, a WSDL description of the web service using known methods (such as the WSDL2Java command line tool). (*Id.*, ¶[0045].)

72. *Huang* further discloses that its generic objects for web components and their attributes are assigned names that are used to relate application components to the generic objects and the generic objects to the vendor-specific objects. For example, *Huang* describes the “Opportunity object type” offered by an exemplary

web-service vendor (Salesforce) and identifies attributes of that web object as “OpportunityName” and “OpportunityId.” (Ex-1005, ¶¶[0044], [0047].)

73. *Huang*, through Figures 3 and 4A-4D and associated text, provides an example of how a user may bind an application component to a web service. In Figure 3, *Huang* shows that the user identifies the web-service vendor and provides the location of the web service via a uniform resource locator:

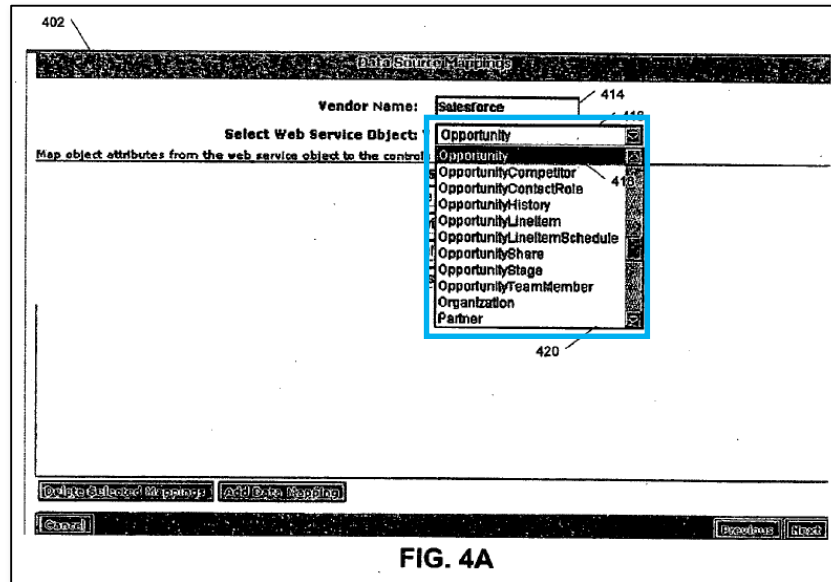


(Ex-1005, Fig. 3.)

Huang explains that “FIG. 3 is an illustrative drawing of defining a web service connection in an application Designer,” where “[a] Connection screen 302 of a web services builder is shown, which allows a user to define parameters for communication with a web service, including a vendor 304 of the web service, [and] a URL 306 of the web service,” among other things. (Ex-1005, ¶[0105].)

74. Figure 4A of *Huang* shows a dropdown menu of the identifiers for various **web service objects** located from the web service vendor and web service

URL described in Figure 3. (Ex-1005, ¶[0106].) In Figure 4A, the user has selected “[a]n Opportunity Object” from “web services object selection menu 420”:



(Ex-1005, Fig. 4A.)

75. Once a user has selected a web service object, the user then selects the **application component** that will be bound to a web service attribute. (See Ex-1005, ¶[0107].) Huang explains that “[a] user can choose an **application component** 420 from the selector 436, which includes a list of components of the application” (*Id.*):

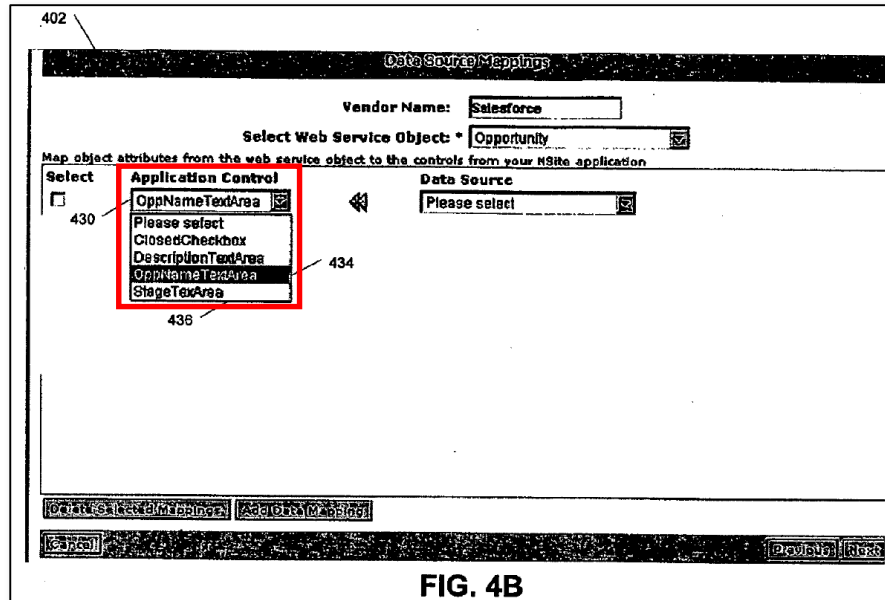
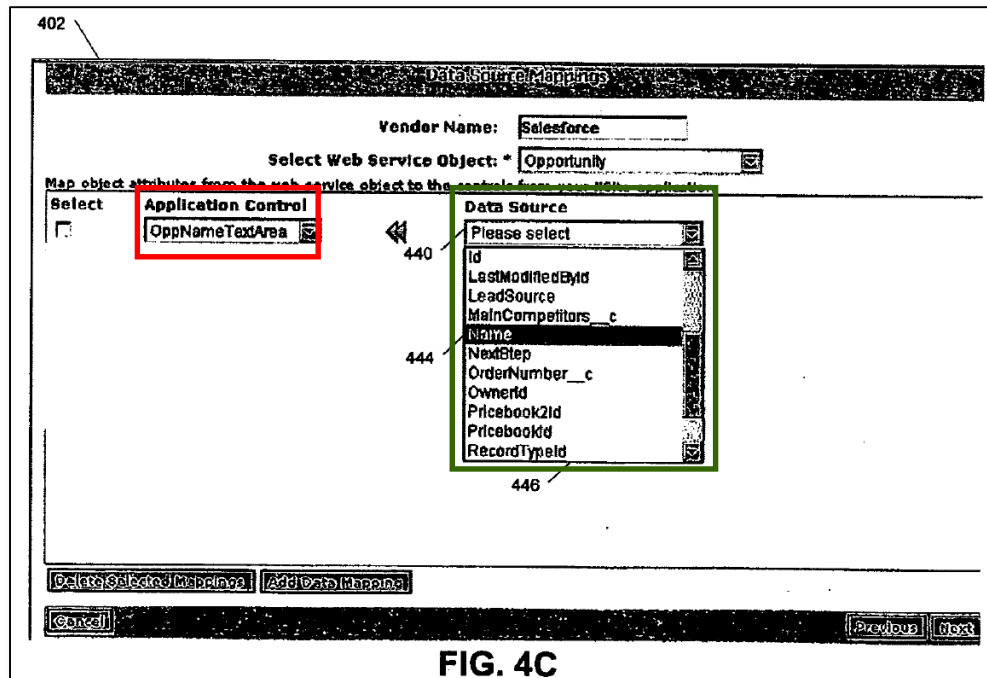


FIG. 4B

(Ex-1005, Fig. 4B.)

76. The user then can choose the **attribute** of the web service object that will be bound to the **application component**. (Ex-1005, ¶[0108].) *Huang* explains that “FIG. 4C shows a web service attribute selector 446 of the data binding screen 402,” where “[a] binding is created by selecting an **application component** and a web service **attribute** using the application component selector 430 and the web service attribute selector 440, respectively” (*Id.*):



(Ex-1005, Fig. 4C.)

77. Figure 4D of *Huang* shows multiple bindings that have been defined between various **application components** (“OppNameTextArea,” “DescriptionTextArea,” “StageTexArea,” and “ClosedCheckbox”) and **attributes** (“Name,” “Description,” “StageName,” “IsClosed”) of the **web service object** (“Opportunity”):

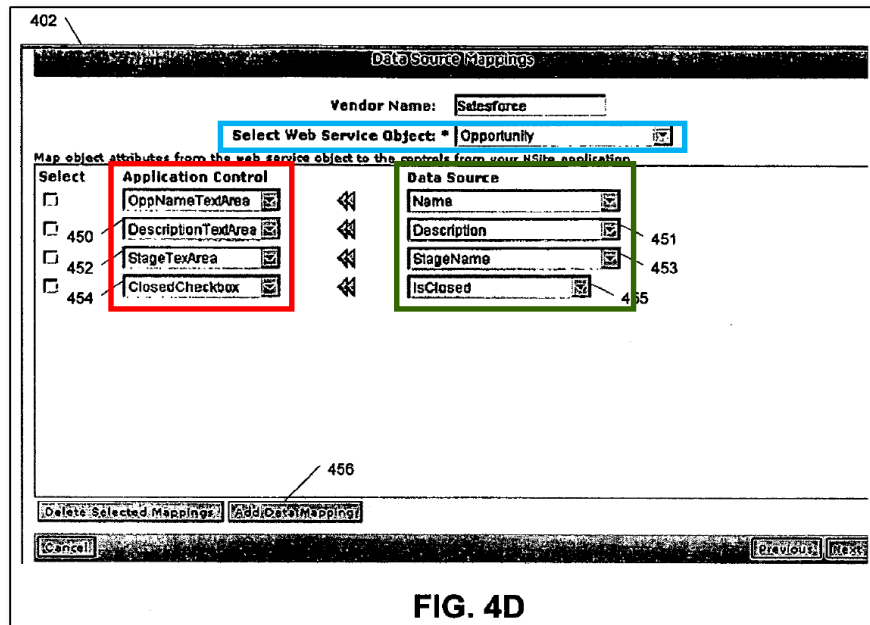


FIG. 4D

(Ex-1005, Fig. 4D.)

78. *Huang* incorporates by reference U.S. Patent Application No. 11/241,073 to Pawan Nachnani et al., which provides additional details regarding *Huang*'s system. (Ex-1005, ¶[0006] (“See commonly owned Patent Application entitled ‘Browser Based Designer and Player,’ filed Sep. 30, 2005, inventors Pawan Nachnani et al., Ser. No. 11/241,073, *incorporated herein by reference in its entirety*, which describes a method of building software applications. The sort of software applications disclosed in that Patent Application may also be used in the system of the present disclosure.”).) *Huang* points to *Nachnani* as describing “[t]he sort of software applications ... [that] may also be used in the system of the present disclosure [of *Huang*],” further explaining that *Huang* discloses the “use of web services in software applications built using the methods described in [*Nachnani*]

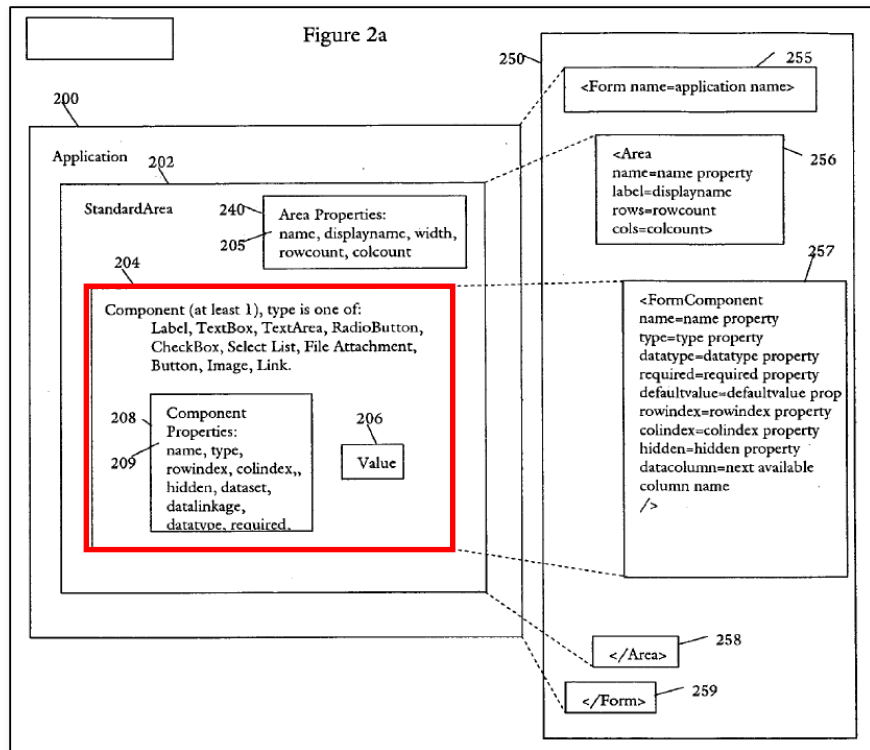
referenced above”; that “[t]he system [of *Huang*] includes a Designer 104 and a Player 106, as described in [*Nachnani*,] referenced above”; and that *Huang*’s “XML definitions are described in [*Nachnani*,]” among other things. (Ex-1005, ¶¶[0006], [0040], [0042].) In my opinion, a POSITA would have understood that *Huang* incorporates by reference the entirety of the disclosures of *Nachnani*, including for the reason that *Huang* expressly and unequivocally states that it is doing so and makes specific reference to *Nachnani* regarding the descriptions of *Huang*’s designer, player, applications, and the method of building *Huang*’s applications. To the extent that *Huang* is deemed not to incorporate *Nachnani* in its entirety, a POSITA would have understood that *Huang* incorporates at least the portions of *Nachnani* describing the player, designer, application, and the method of building the application. The ’073 application published as U.S. Patent Application Publication No. 2007/0079282 (“*Nachnani*”).

79. *Nachnani* explains that the “Player UI ... executes an application 135” and “presents the application 135 to a user 130 via the web browser.” (Ex-1006, ¶[0040].) Figure 8 of *Nachnani* further shows that the “Player UI” operates on a “Client Computer,” and operates to “present[] user interface portions of the clients to a user.” (*Id.*, ¶[0097] & Fig. 8.)

80. *Nachnani* further elaborates on the code and operation of the player and application. *Nachnani* confirms that an application is specified or defined “using

Extensible Markup Language (XML).” (*Id.*, ¶[0045]; *see also id.*, ¶¶[0052], [0056], [0063], [0103]-[0104].) *Nachnani* also confirms that a “web browser 832 ... run[s] an Application Player client 833” and that “[a] user interacts with the Application Player client 833 to provide data to an application and view data associated with the application.” (*Id.*, ¶[0097].) *Nachnani* further states that “[t]he Player Client 833 allows users to load and execute application definitions and save and load application data.” (*Id.*) The Player Client 833 “presents user interface portions of the clients to a user, executes any code associated with the clients, e.g. JavaScript code, allows the user to interact with the client user interfaces, and sends any results of the user interaction to the server.” (*Id.*)

81. *Nachnani* provides further details regarding the **application components** utilized in *Huang*. Figure 2A from *Nachnani* is reproduced below:



(Ex-1006, Fig. 2A.)

82. *Nachnani* explains that “[a]n application 200 may include at least one StandardArea 202. A StandardArea 202 includes area properties 240 and at least one component 204.” (Ex-1006, ¶[0041].) *Nachnani* further explains that “component 204 is a user interface element of a specific type,” such as “a Label, Text Box, Text Area, Radio Button, Check Box, Select List (e.g. a drop-down list), File Attachment, button, Image, or link,” and “has associated properties,” which are shown in the table below (Ex-1006, ¶[0042]):

| COMPONENT TYPE | DESCRIPTION | PROPERTIES |
|----------------|--|---|
| Text Box | A text input field for reading data of a specific type, e.g. string or numeric data. Contains a text value. | Name, Required, Readonly, Keyword Field, Data Type, Allowed Characters, Visibility Privileges, Edit privileges, Routing edits (enable or disable), data type, data format, max characters, default value, action (event handler). |
| Text Area | A text edit box for editing a text string. Can display multiple rows of text and allow the user to edit the text. Contains a text value. | Name, required, read only, keyword field, visibility and edit privileges, routing edits (enable or disable), allowed characters, data format, max characters, max rows, max chars, default value, action (event handler). |
| Date Picker | A data selector. Contains a date value. | Name, required, read only, keyword field, visibility and edit privilege, routing edits (enabled or disabled), data format, default value, action (event handler). |
| Check Box | A check box with two possible values (checked or unchecked). Contains a boolean value. | Label, name, edit and visibility privileges, routing edits (enable or disable), initial state (checked or unchecked), action (event handler). |
| Radio Buttons | A set of buttons, one of which can be selected. Contains a value representing a selected button. | Name, radio group, edit privileges, routing edits, initial state, checked value, action (event handler). |
| Combo Box | A drop-down box for selecting one item from a list of items. | Name, required, keyword field, visibility and edit privileges, routing edits (enable or disable), data type, height, allow multiple selections (yes or no), data lookup source, action (event handler). |
| Label | A text label for displaying information. Does not contain a value. | Name, layout style, alignment. |
| Button | A button that can be selected by a user to cause an event handler to be called. | Label, name, visibility and edit privileges, action (event handler), web service, link. |
| Attachment | A file associated with the application. | Name, attachment type, visibility privileges, action privileges. |
| Link | A web link that refers to a web page, web service, or other resource. When the link is selected in the Player UI, the web page, web service, or other resource will be accessed. | Label, name, visibility and edit privileges, action (event handler), web service, link (URL). |

(Ex-1006, ¶[0042].)

Nachnani explains that “[t]he component properties 208 include a name property, which uniquely identifies the component within the area 204, a type property, e.g. label or text box, a label property, the value of which is displayed in the user interface for some component types, such as the label component type.” (Ex-1006, ¶[0043].)

Nachnani continues that the “component properties 208 also include rowindex and

colindex properties, which indicate the position of the component within the area, a hidden property, which indicates whether the component is to be displayed or hidden in the Player UI, and a datalinkage property, which indicates a source of values for the component, e.g. a database column containing values for a selection list, a datatype property, which indicates the permitted type of the value 206, and a required property, which indicates whether a value is required for the component 204.” (*Id.*)

83. *Nachnani* also describes that these components “may include a link” and that “the link may refer to a web service, and the client program may receive the data value from the web service in response to a user action.” (Ex-1006, ¶[0019].) *Nachnani* explains that “[t]he link 142 may send or receive the data value 142 to or from the application 146, the web page 148, or the web service 150, in response, for example, to input from the user 130.” (*Id.*, ¶[0040]; *see also id.*, ¶¶[0087] (“Applications created using the Application Designer can invoke Web Services when executed in the Application Player. In a typical Web services scenario, an application sends a request to a service at a given URL using the Simple Object Access Protocol (SOAP) over HTTP. The service receives the request, processes it, and returns a response.”), [0088] (“The Web Service may provide a value for the component, or, vice versa, the component may provide a value for the Web Service.”).)

B. *Shenfield*

84. U.S. Patent Application Publication No. 2006/0200749 to Michael Shenfield (“*Shenfield*”) is titled “System and Method for Conversion of Generic Services’ Applications into Component Based Applications for Devices.” (Ex-1007, Cover (54).) *Shenfield*, in relevant part, addresses web applications capable of providing access to web services to the then-emerging class of Internet-accessible devices, such as mobile phones and other devices with wireless communication capabilities. (*Id.*, ¶[0002].)

85. *Shenfield* recognizes that devices like cell phones generally have restricted resources and less computing power than, for example, personal computers, creating a tension between resource intensive solutions that promote interoperability (e.g., solutions that hinder data persistence and require rendering screens at runtime) and device capabilities. (Ex-1007, ¶¶[0002]-[0003].) *Shenfield* explains that certain solutions “have the advantage of being adaptable to operate on a cross-platform basis for a variety of different devices, but have a disadvantage of requesting pages (screen definitions in HTML) from the Web Service, which hinders the persistence of data contained in screens” and that “screens are rendered in runtime, which can be resource intensive.” (Ex-1007, ¶[0003].) On the other hand, “[n]ative applications have the advantage of ... providing a relatively optimized

application program for each runtime environment” but “have disadvantages of not being platform independent.” (*Id.*)

86. In relevant part, *Shenfield* discloses representing “a compiled web application 107 into a series of descriptors using a selected structured definition language (e.g. XML),” which the device receives as a “series of the interactively defined components 400, 402, 404, 406,” and executes on a device “to operate as the client application 105 for communication with the web service(s) through messaging.” (*Id.*, ¶¶[0022]-[0023].) *Shenfield* explains that “data components 400 define entities which are used by the component application program 302.” (Ex-1007, ¶[0041].) *Shenfield* explains that presentation components “define the appearance and behavior of the component applications 105 as it is displayed by the user interface 202.” (Ex-1007, ¶[0043].) *Shenfield* also explains that workflow components define “presentation workflow and message processing,” and are “written as a series of instructions” that can “support[] a correlation between the messages and defines application flow as a set of rules for operations on the other components.” (Ex-1007, ¶[0045].) *Shenfield* further explains that its “component applications 105 may be executed as native code or interpreted by another software module or operating system on the device 100.” (Ex-1007, ¶[0029].)

87. *Shenfield* further explains that these components may be device-independent and apply across different devices or platforms but may also utilize

platform-specific elements suited for particular runtime environments. *Shenfield* states that the above-described components “can be hosted in a Web Service 106 registry in a metadata repository 700 (see FIG. 1) as a bundle of platform-neutral data 400, message 404, workflow 406 component descriptors with a set of platform-specific presentation component 402 descriptors for various predefined client runtimes (i.e. specific runtime environments 206—see FIG. 2).” (*Id.*, ¶[0044].) *Shenfield* also states that workflow components, “similar to the multiple presentation components 403a, 403b, 403c, can define differing work flows based upon different supported capabilities or feature of particular devices 100.” (*Id.*, ¶[0045].)

88. By utilizing both device-independent and device-dependent elements, *Shenfield* achieves its objective of “application programs, other than page-based applications, that can be run on client devices having a wide variety of runtime environments, as well as having a reduced consumption of device resources.” (Ex-1007, ¶[0005].) Specifically, “the component application 105 model combines the simplicity of browser-based applications with the efficiency of native application execution.” (*Id.*, ¶[0054].)

X. THE PRIOR ART DISCLOSES OR SUGGESTS ALL OF THE FEATURES OF THE CHALLENGED CLAIMS

A. Grounds Challenging Claims 1, 3, 5-7, 11-12, 14, 16-18, and 22 of the '755 Patent

89. I understand that the Petition is challenging claims 1, 3, 5-7, 11-12, 14, 16-18, and 22 of the '755 patent on one ground.

90. In **Ground 1**, I explain how claims 1, 3, 5-7, 11-12, 14, 16-18, and 22 of the '755 patent are obvious over the system in *Huang* disclosing a designer and player for browser-based web services applications as modified by utilizing a web services registry and implementing device-dependent components as a runtime player, as disclosed in *Shenfield*. I further explain that a POSITA would have been motivated to make such modifications, especially in view of the needs of emerging Internet-accessible devices, and that a POSITA would have had a reasonable expectation of success due to, among other things, the similarities in the underlying technologies of *Huang* and *Shenfield*.

B. Ground 1: *Huang* in View of *Shenfield* Render Claims 1, 3, 5-7, 11-12, 14, 16-18, and 22 of the '755 Patent Obvious

91. I have reviewed *Huang* and *Shenfield*, and it is my opinion that *Huang* in view of *Shenfield* discloses or suggests, and therefore renders obvious, all of the features recited in claims 1, 3, 5-7, 11-12, 14, 16-18, and 22 of the '755 patent.

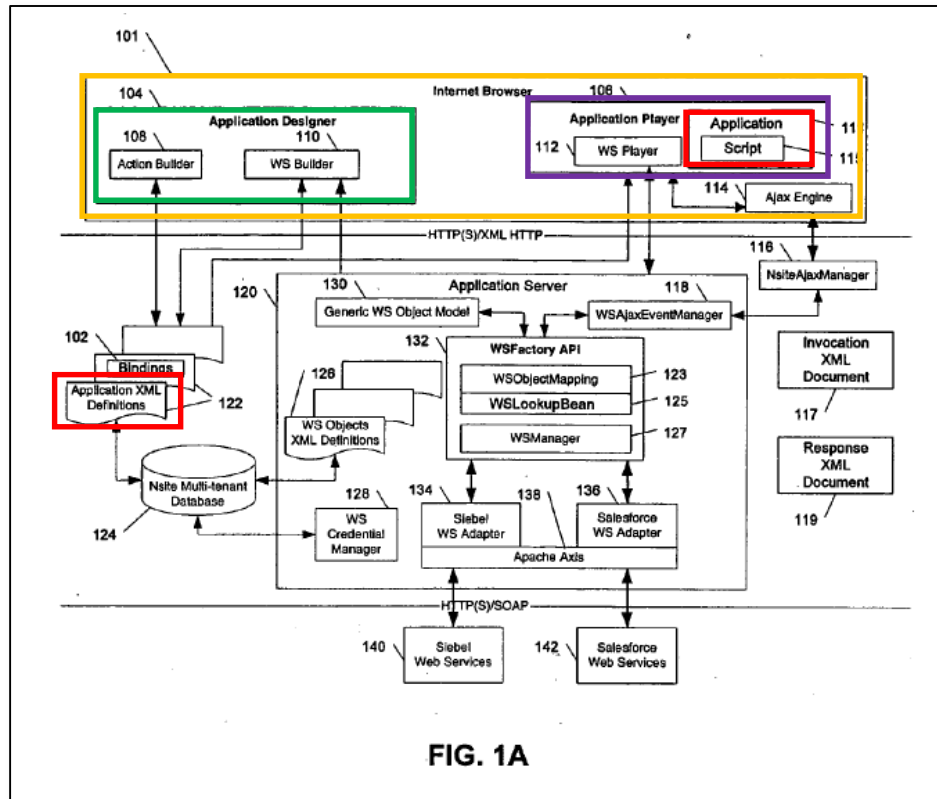
1. Claim 1

a. [1.pre] “A system for generating code to provide content on a display of a device, said system comprising:”

92. I understand that the phrase “[a] system for generating code to provide content on a display of a device, said system comprising:” is the preamble to claim

1. I have been asked to assume that the preamble is a claim limitation. Under that assumption, it is my opinion that *Huang* discloses this limitation.

93. *Huang* expressly discloses a *system*. Specifically, *Huang* states that “FIG. 1A is an illustrative drawing of a computer enabled *system* for *developing* and executing *browser-based software applications*.” Figure 1A of *Huang*, reproduced below, shows a *system* that includes **designer 104** (which executes within **web browser 101**) that generates **Application XML Definitions 122** which are loaded into **player 106** as **application 111**. **Player 106** also executes within one or more **web browser[s] 101**:



(Ex-1005, Fig. 1A; *see also id.*, ¶¶[0040]-[0043].)

94. It is my opinion that a POSITA would have understood that *Huang* discloses that its *system generates code*. *Huang* explains that “[e]ach element or component shown in FIG. 1A is a software computer program entity to be executed on a computer” (Ex-1005, ¶[0040].) and that Designer 104 includes “a wizard ... which creates the binding or mapping of meta-information to generate composite or other software applications.” (Ex-1005, ¶[0008].) From these disclosures that *Huang*’s system generates application 111, which is a “software computer program entity to be executed on a computer,” and that Designer 104 “generate[s] composite or other software applications,” a POSITA would have understood that *Huang*’s

designer generates computer code because software that executes on a computing device is, at bottom, written in code.

95. *Huang* further discloses that Designer 104 generates applications in XML format. (Ex-1005, ¶[0040].) A POSITA would have understood that XML is a type of computer code and, therefore, Designer 104 generates computer code in the form of XML applications.

96. It is also my opinion that *Huang* discloses that this code *provides content on a display of a device*. As discussed below, *Huang*'s disclosures would have been understood to demonstrate that *Huang*'s player presents a user interface for content defined by *Huang*'s applications.

97. For example, and as discussed more fully below, *Huang* discloses that “Player 106 is a computer program that provides a user interface which executes a browser based application 111, which can receive values from web services, send values to web service, and invoke other operations provided by web services.” (Ex-1005, ¶[0040]) A POSITA would have understood that “provid[ing] a user interface” involves displaying content. A “user interface” in the computing arts generally refers to how a user and a computer system interact. In the context of *Huang*, which relates to a “computer program that provides a user interface” as previously mentioned, a POSITA would have understood that the software presents

visual elements that present the user with data that application 111 obtains from a web service or allow the user to input values which is sent to the web service.

98. *Huang* also discloses that “Player 106 execute[s] in a conventional Internet Browser 101, which may be, for example, Microsoft Internet Explorer or Mozilla Firefox™.” (Ex-1005, ¶[0040].) A POSITA would have understood that an Internet Browser is a software component intended to visually display HTML pages to a user. In the context of *Huang*’s Player, which as discussed above is a “computer program that provides a user interface,” and applications, which are “browser-based applications,” a POSITA would have understood that the fact the Player is being executed in a web browser to provide a user interface for browser-based applications means that it uses the browser and player to visually display the user interface for the application.

99. Further, *Nachnani*, which *Huang* incorporates by reference (Ex-1005, ¶[0006]), further confirms that *Huang* discloses a method for displaying content on a display of a device. *Nachnani* explains that the “Player UI ... executes an application 135” and “presents the application 135 to a user 130 via the web browser.” (Ex-1006, ¶[0040].) Thus, a POSITA would have understood that *Nachnani* confirms that the “Player UI” provides a visual display through the web browser of the content defined in the application. Figure 8 of *Nachnani* further

shows that the “Player UI” operates on a “Client Computer,” and operates to “present[] user interface portions of the clients to a user.” (*Id.*, ¶[0097] & Fig. 8.)

- b. [1.a] “computer memory storing a registry of: a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and b) the address of the web service”**

100. It is also my opinion that, to the extent *Huang* does not expressly disclose a registry, a POSITA would have found it obvious to modify the system of *Huang* to include *a computer memory storing a registry of symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service and the address of the web service.*

101. As discussed above, I am informed that the district courts construed the term “registry” to mean “a database that is used for computing functionality,” the term “web component” to mean “software objects that provide functionalities of a web service”, “web service” to mean “a software system that supports interaction between devices over a network”, and “for evoking one or more web components” to mean “for calling up one or more web components.” In my opinion, *Huang* in

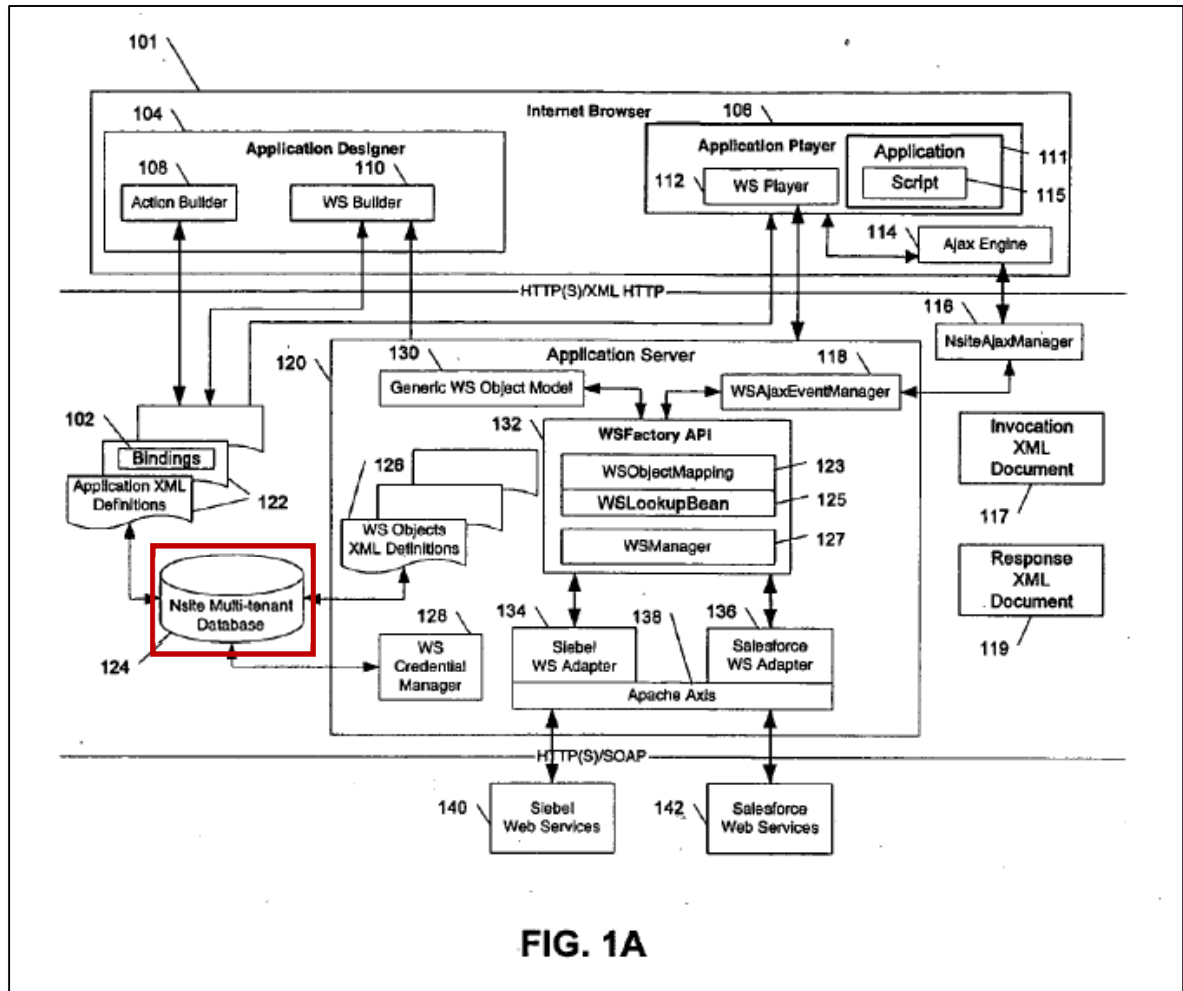
view of *Shenfield* renders limitation 1.a obvious under the district courts' constructions of these terms.

102. For clarity, I have attempted to address each aspect of element 1.a separately in my analysis below.

(i) “computer memory storing a registry of”

103. In my opinion, *Huang* in view of *Shenfield* render *a computer memory storing a registry* obvious. As discussed below, to the extent *Huang* does not expressly disclose a registry, a POSITA would have found it obvious to modify *Huang*'s system to include a registry, such as the registry disclosed in *Shenfield*.

104. *Huang* discloses *a computer memory*. For example, *Huang* recites that “[a]pplications created by the Designer 104 are stored in Extensible Markup Language (XML) format as XML definitions 122 in **a database 124**, and loaded into the Player 106 as the application 111.” (Ex-1005, ¶[0042].) Figure 1A is reproduced below, showing **database 124**.

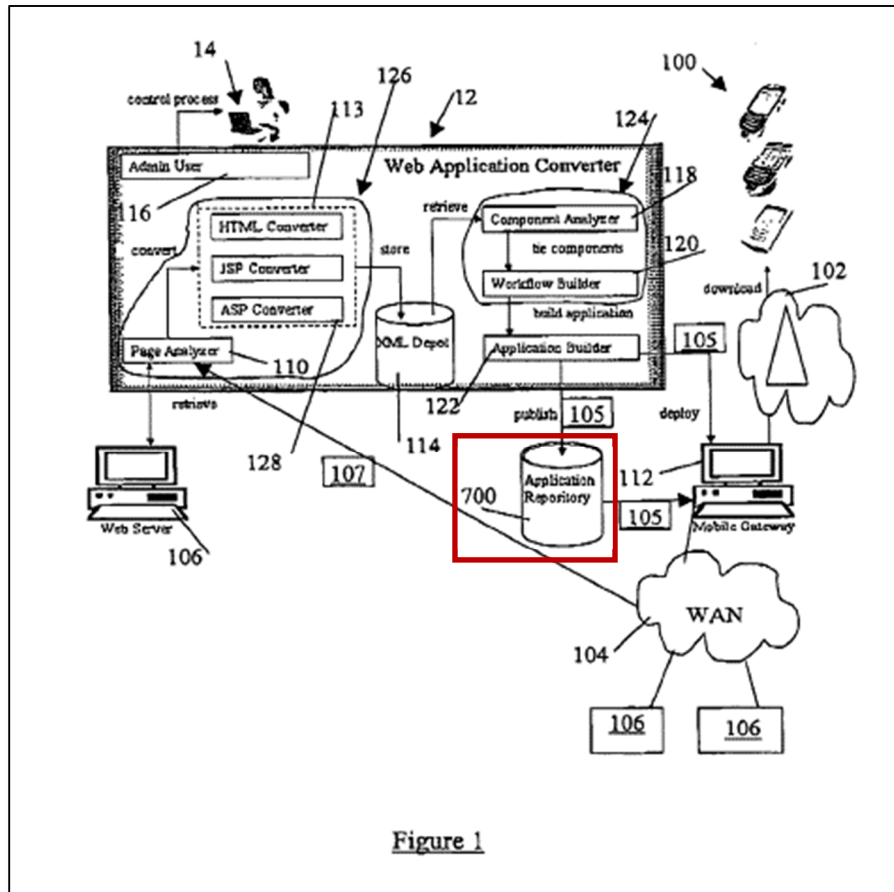


(Ex-1005, Fig. 1A.)

105. A POSITA would have understood that *Huang's* database 124 is a computer memory. *Huang's* disclosure that database 124 stores computer code in the form of XML definitions of *Huang's* applications (Ex-1005, ¶[0042]; *see also id.* ¶[0040] (identifying all components in Figure 1A, including “applications,” as “software computer program entit[ies]”)) indicates that database 124 serves as a computer memory that is a repository of computer data, which in this case are the XML definitions of *Huang's* applications.

106. *Shenfield* expressly discloses web services *registries*. Specifically, *Shenfield* explains that “web service 106 can be defined as a software service, which can implement an interface such as expressed using Web Services Description Language (WSD) registered in a Universal Discovery Description and Integration (UDDI) services registry.” (Ex-1007, ¶[0025].) The UDDI registry is a well-known registry that provides listings of information necessary to connect a web service, such as its network location, its capabilities, and communication semantics. (*E.g.*, Ex-1019, ¶[0006].) *Shenfield* further states that “[t]he application definition of the components 400, 402, 404, 406 of the component applications 105 can be hosted in a Web Service 106 registry in a metadata repository 700 (see FIG. 1) as a bundle of platform-neutral data 400, message 404, workflow 406 component descriptors with a set of platform-specific presentation component 402 descriptors for various predefined client runtimes (i.e. specific runtime environments 206—see FIG. 2).” (Ex-1007, ¶[0044].) A POSITA would have understood that the “registr[ies]” disclosed in *Shenfield*, such as the UDDI registry, are structured sets of data (i.e., databases) relating to computer functionality—in this case, accessing web services. (*See, e.g.*, Ex-1018, 90 (“The [UDDI] specifications offer users a unified and systematic way to find service providers through a centralized registry of services that is roughly equivalent to an automated online ‘phone directory’ of web services.”).)

107. It is also my opinion that *Shenfield* discloses a similar, analogous *computer memory*, on which a *registry is stored*. *Shenfield* also explains that “[t]he application definition of the components 400, 402, 404, 406 of the component applications 105 can be hosted in a Web Service 106 registry in a metadata repository 700 (see FIG. 1) as a bundle of platform-neutral data 400, message 404, workflow 406 component descriptors with a set of platform-specific presentation component 402 descriptors for various predefined client runtimes (i.e. specific runtime environments 206—see FIG. 2).” (Ex-1007, ¶[0044].) Figure 1 from *Shenfield* is reproduced below, identifying **metadata repository 700**, which is also referred to as “**Application Repository 700**”:



(Ex-1007, Fig. 1.)

108. Thus, *Shenfield* discloses that a registry can contain the information necessary to access a web service and a POSITA would have understood that the registry can be stored on a computer memory similar to *Huang*'s database 124 because *Huang*'s database 124 shares a number of parallels with *Shenfield*'s application repository 700. Specifically, *Shenfield* discloses that its application components are stored in application repository / metadata repository 700 much like *Huang* stores the XML definitions of its applications in database 124 (Ex-1005, ¶[0042]; Ex-1007, ¶[0044]) and, therefore, store the same type of data. Both

memory stores also function similarly, receiving data from the system (*Huang's* applications or *Shenfield's* component applications) and provide them to the end-user. (See Ex-1005, Fig. 1a; Ex-1007, Fig. 1.) Further, both computer memories are arranged and connected within the system in a substantially similar manner—both are identified as stand-alone components in communication with both the system and end-user. (See Ex-1005, Fig. 1a; Ex-1007, Fig. 1.) Accordingly, these are analogous computer memories where a POSITA would have found modifying *Huang's* system to store a registry within database 124 obvious. I refer to my analysis below regarding the motivation to combine *Shenfield's* registry with *Huang's* system for additional analysis. (*Infra* Section X.B.1.b(v).)

- (ii) “[a registry of]: a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network”

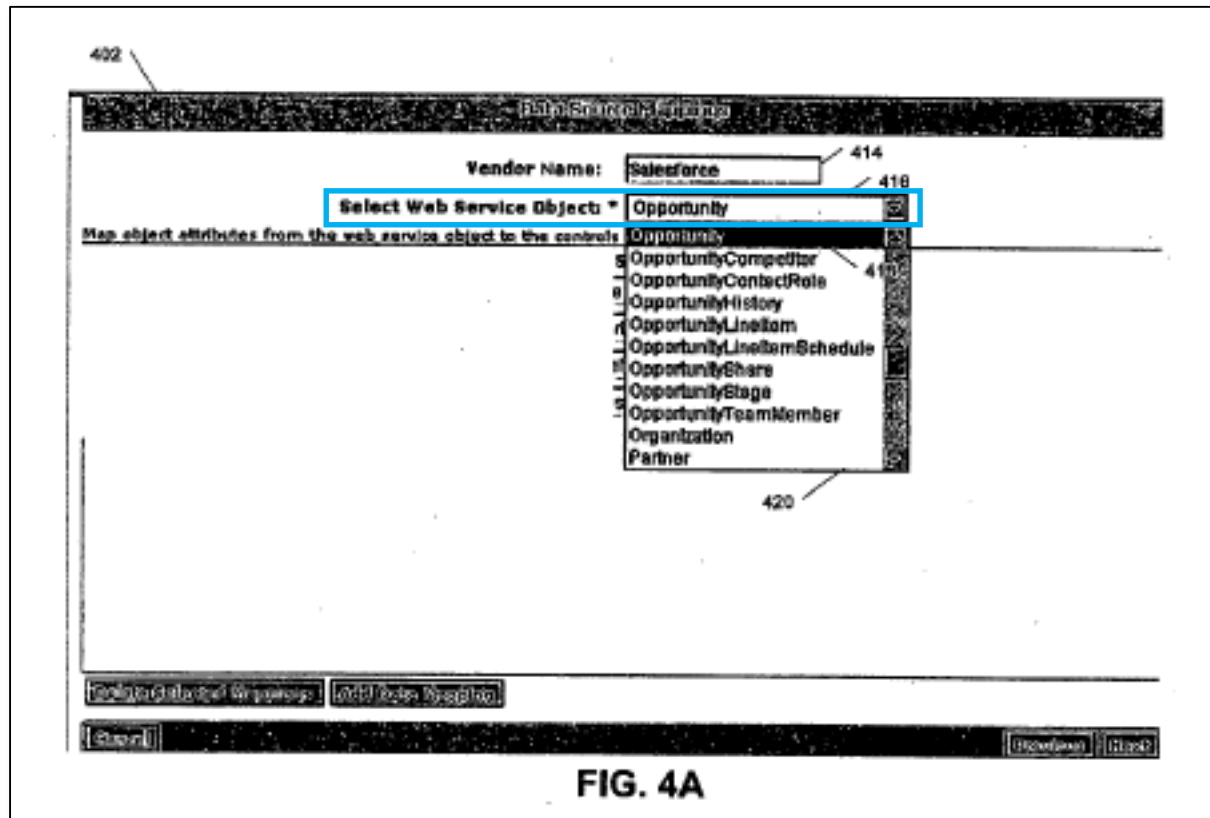
109. Element 1.a of claim 1 also requires that the registry include “symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network.” (Ex-1001, Claim 1.) As noted above, the district courts construed the term “web component” to mean “software objects that provide functionalities of a web service,” “web service” to mean “a software system that supports interaction between devices over a network,” and the phrase “for evoking one or more web components” as “for calling up one or more web components.” (*Supra* Section VII.) In my opinion, *Huang* in view of

Shenfield renders a registry of *symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network* obvious under the district courts’ construction.

110. In my opinion, *Huang* discloses “web services” as construed by the district courts (“a software system that supports interaction between devices over a network”). *Huang* explains that “web services” are “computer application (software) component[s] accessible over open protocols.” (Ex-1005, ¶[0003].) And *Huang* continues that “[w]eb services are intended for purposes of interoperability, for instance for enterprise application integration and business-to-business integration.” (*Id.*) A POSITA would have understood that these disclosures indicate that the “web services” disclosed in *Huang* are software systems that support interaction between devices over a network. *Huang* confirms this understanding, further stating that “[w]eb services are web-based enterprise applications that use open XML standards and transport protocols to exchange data with calling clients (programs).” (*Id.*, ¶[0005].)

111. In my opinion, *Huang* discloses *web components*. For example, *Huang* discloses a “list of selected web services object instances” (Ex-1005, ¶[0109]) and a “list of objects provided by a web service” (*id.*, ¶[0043]), which are the individual services obtainable from a vendor. (See also *id.*, ¶[0109].) Figure 4A, reproduced

below, shows a list of **web service objects** from which a user may select a web object.



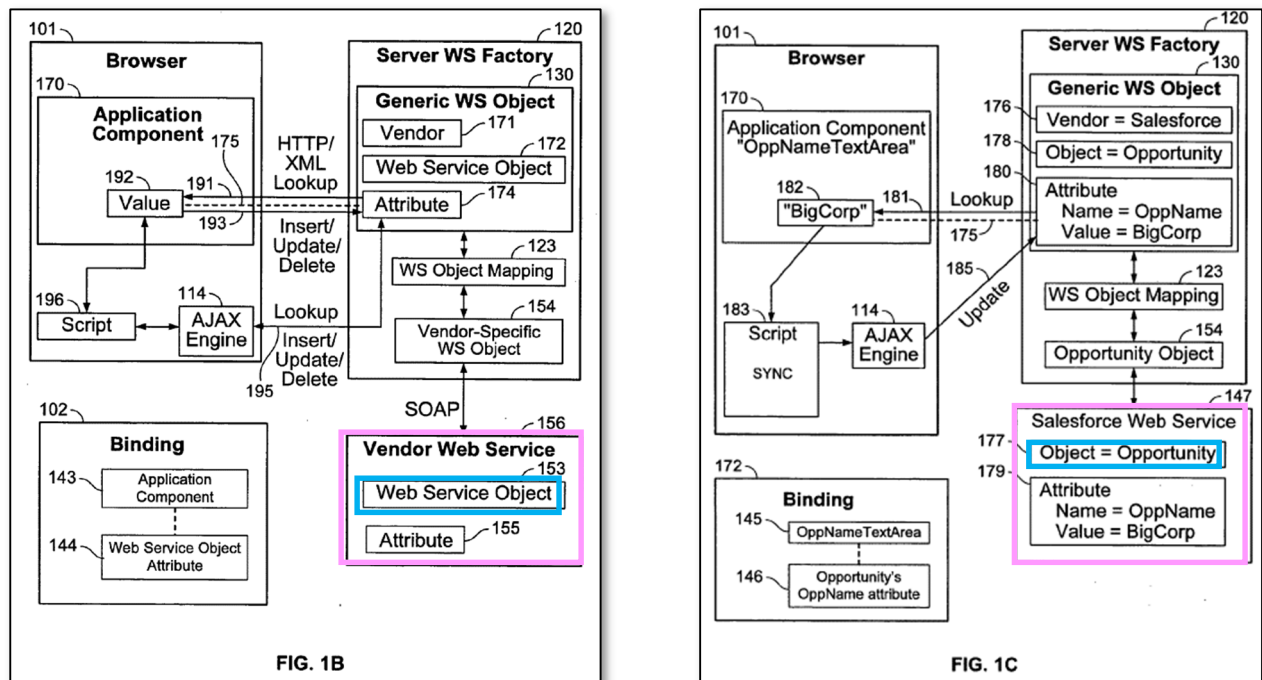
(Ex-1005, Fig. 4A.)

In this example, the user has selected the “Opportunity” object from the Salesforce web service. Figure 4A shows that the Salesforce web service, at least in this example, contains multiple separate services and thus multiple web objects (*web components*).

112. A POSITA would have understood that the “objects” disclosed in *Huang* are web components. The list of “objects” obtained from a web service “include data and operations (logic), in another organization’s computer system.”

(Ex-1005, ¶[0007].) Accordingly, a POSITA would have understood that these objects are individual functionalities offered by a web service that carry out specific tasks within that web service.

113. Further, Figures 1B and 1C, reproduced below, further indicate that Huang's **web objects** are individual functionalities within a suite of **web services** offered by a vendor.

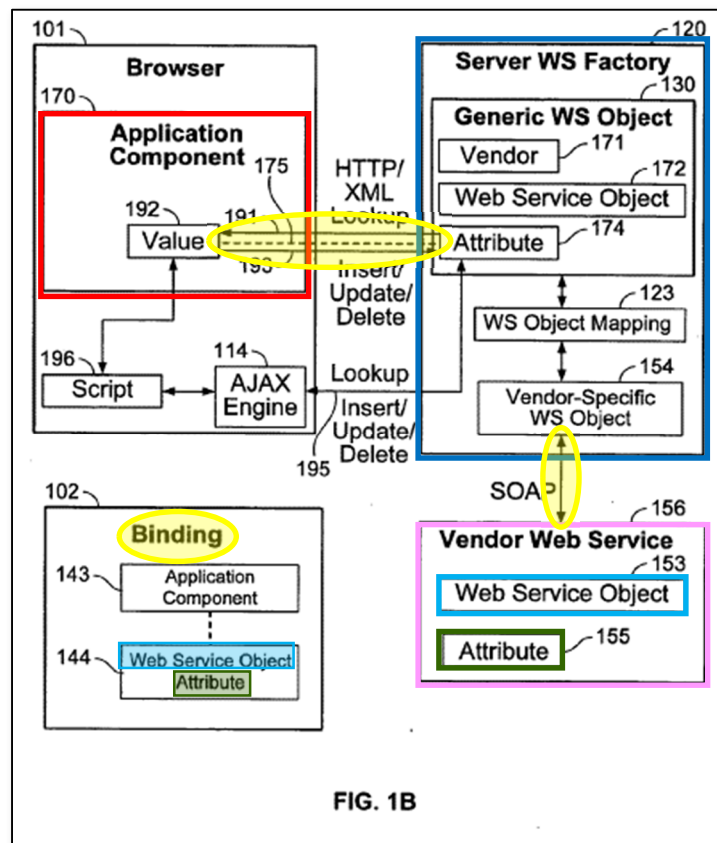


(Ex-1005, Figs. 1B, 1C.)

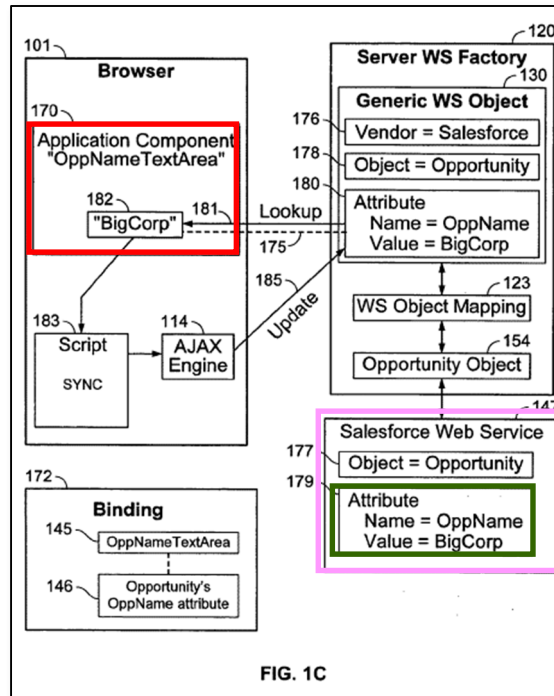
As can be seen in the figures, **web service object 153** and the **“Opportunity” object 177** are an individual element of what is offered by the **vendor web service 156** or the **Salesforce Web Service 147**.

114. It is also my opinion that *Huang* discloses that its web objects (which a POSITA would have understood are the claimed *web components*) are *each related to inputs and outputs of the web service*. *Huang* states that each web service object has attributes. (Ex-1005, ¶[0047] (“[W]eb service data is represented as objects with attributes.”).) *Huang* further explains that a user may “define bindings 102 between **components of the application**, e.g. buttons and tables, and **web service attributes**, e.g. values that can be stored in or retrieved from web services.” (*Id.*, ¶[0055].) More specifically, *Huang* explains that “[t]he binding 193 associates a value 192 of an **application component 170** with an **attribute 155** of a **web service 156**” and that “[t]he binding is a two-way mapping which specifies that a particular **web service attribute 155** can be set to the value 192 of the **application component 170**, and, conversely, that the value of the **application component 170** can be set to the value of the **attribute 155**”—i.e., **application component 170** can be an input that sends a value to **web service 156** and sets the corresponding value of the web service (**attribute 155**) equal to the value of the **application component 170**, or the **application component 170** can receive an output of the **web service 156** by the **application component 170** being set to a value equal to what is in the corresponding value of the web service (**attribute 155**). (*Id.*, ¶[0056]; *see also id.*, ¶[0057] (“setting and getting the value of a **web service attribute 155** associated with a **web service object 153**”).) This two-way flow of data between value 192 of

application component 170 and **attribute 155** via **server 120** can be seen in, for example, Figure 1B reproduced below.



115. For example, *Huang* discloses in one example that “there could be an object named Opportunity with attributes named OpportunityName and OpportunityId.” (Ex-1005, ¶[0047].) *Huang* further explains that Figure 1C shows that “binding 175 is represented as an association between a **component 145**, named OppNameTextArea, and a **web service attribute 146**, named OppName,” which is a part of the **Salesforce web service 147**. (*Id.*, ¶[0068].)



(Ex-1005, Fig. 1C.)

116. *Huang* also explains that Appendix D (Ex-1005, ¶[0139]) shows an example XML element format for attribute bindings (Ex-1005, ¶[0059]):

[0139]

APPENDIX D

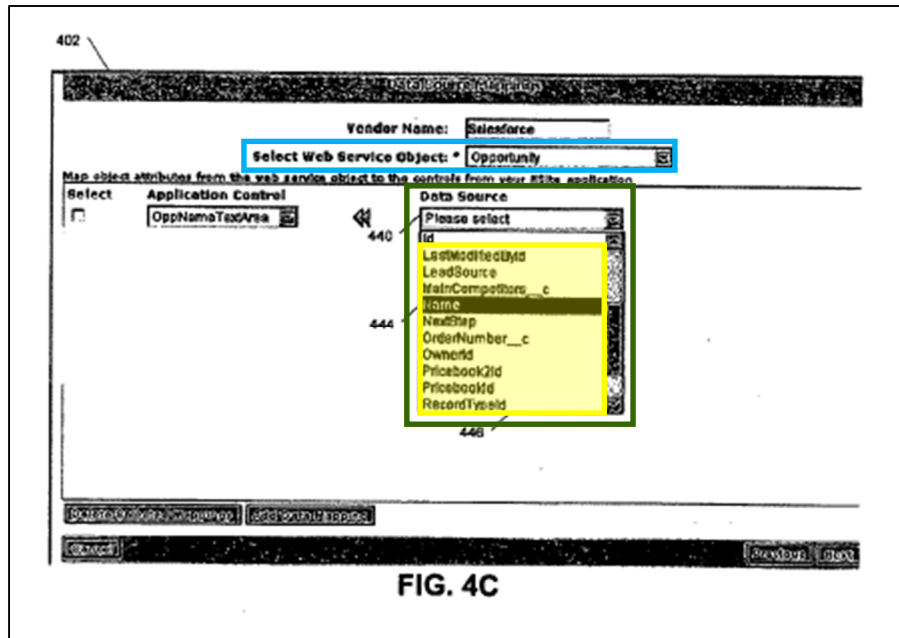
```
<WebServiceBinding vendormame="Salesforce"
  objectname="Opportunity"
  username="" wsurl="https://www.salesforce.com/services/Soap/ui/6.0 ">
  <WSAttributeBindingList>
    <WSAttributeMapping formentity="OppNameTextArea"
  wsattribute="OppName"/>
    <WSAttributeMapping formentity="DescriptionTextArea"
  wsattribute="Description"/>
    <WSAttributeMapping formentity="StageTex.Area"
  wsattribute="StageName"/>
    <WSAttributeMapping formentity="ClosedCheckbox"
  wsattribute="IsClosed"/>
  </WSAttributeBindingList>
  <WSFilterBindingList>
    <WSFilterField filterfieldwsattribute="OppName"
  formentity=""
  filterfieldtype="Dropdown"/>
    <WSFilterField filterfieldwsattribute="CloseDate"
  formentity=""
  filterfieldtype="TextBox"/>
  </WSFilterBindingList>
  <WSTableBindingList>
    <WSLookupTableColumn wsobjattribute="OppName"
  label="Name" sortable="false"/>
    <WSLookupTableColumn wsobjattribute="Description"
  label="Description" sortable="false"/>
    <WSLookupTableColumn wsobjattribute="CloseDate"
  label="CloseDate" sortable="false"/>
    <WSLookupTableColumn wsobjattribute="CreatedById"
  label="CreatedById" sortable="false"/>
    <WSLookupTableColumn wsobjattribute="CreatedDate"
  label="CreatedDate" sortable="false"/>
    <WSLookupTableColumn wsobjattribute="IsClosed"
  label="IsClosed" sortable="false"/>
  </WSTableBindingList>
</WebServiceBinding>
```

(Ex-1005, ¶[0139].)

117. In view of *Huang*'s discussion of attributes (which are the *inputs and outputs of a web service*) being part of a web object (the claimed *web component*) (Ex-1005, ¶[0047]) and that application components are bound to attributes such that the application component can get or set a value of the web service (*id.*, ¶¶[0055]-[0057], [0059], [0068], [0139], Figs. 1B-1C), a POSITA would have understood that *Huang*'s web objects (*web components*) are associated with and relate to *inputs and outputs of a web service* in the form of attributes.

118. It is also my opinion that *Huang* discloses that the identifiers for its web objects (which are the claimed *web components*) and their attributes (the claimed

inputs and outputs of a web service) are identified by *symbolic names*. As discussed above, *Huang* discloses that each web object (e.g., “Opportunity”) are web components that are associated with a plurality of attributes (e.g., “OpportunityName” and “OpportunityId”), which are the inputs and outputs of the web service. (E.g., Ex-1005, ¶[0047]; *supra* Section X.B.1.b.) *Huang* provides an example describing the “Opportunity” **web object** from the Salesforce web service as having a list of “Data Sources,” which *Huang* explains are its **attributes**. (Ex-1005, ¶[0108] (“Note that the web service attribute is referred to as a Data Source in FIG. 4C.”); *see also id.* ¶[0106] (“An Opportunity object 418 has been selected by the user.”), [0107] (“FIG. 4B shows a data binding screen 402, which allows a user to establish binding between application components (also referred to herein as controls), and as attributes of the selected web service object 416 that was selected in the screen of FIG. 4B.”).)



(Ex-1005, Fig. 4C.)

119. It is my opinion that a POSITA would have understood that the names of a web object, such as “Opportunity,” and the names of attributes (or “Data Sources”), such as “Name” and “NextStep” shown in Fig. 4C, are *symbolic names*. Each of these names are a series of symbols/character strings, which in these examples are letters of the alphabet forming descriptors that can be understood by humans.

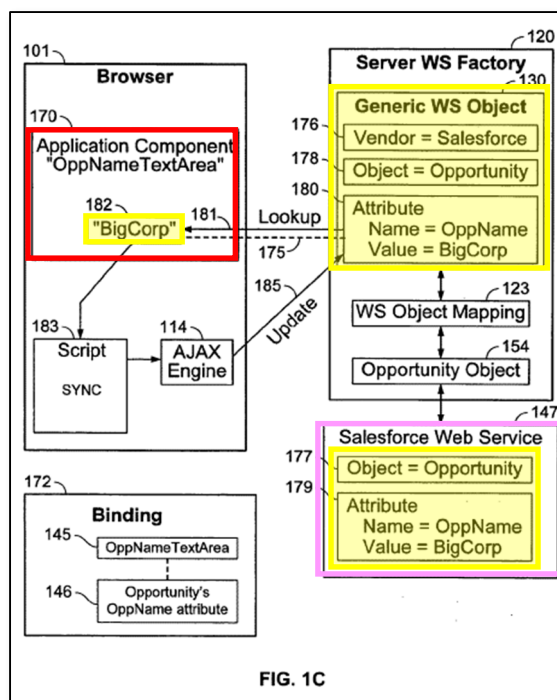
120. It is also my opinion that *Huang* explains that these *symbolic names* are *required for evoking or calling up one or more web components each related to a set of inputs and outputs of a web service*. *Huang* discloses that application components, which are bound to the *symbolic names* of *Huang*’s web objects (*web components*) and attributes (*inputs and outputs of a web service*) (*supra* X.B.1.b),

“can be configured to invoke a web service” (Ex-1005, ¶[0073]) and that *Huang*’s system utilizes “generic web services object[s]” to “call vendor-specific objects” (*id.*, ¶[0074]). (*See also id.*, ¶¶[0049] (explaining the system can “perform web service invocations in response to user requests”), [0067] (a “script 115 [that] is associated with a component ... can also invoke web services”).) More specifically, *Huang* explains that its symbolic names are elements of a generic object class and that these generic objects serve as necessary intermediaries between *Huang*’s application components and web service objects to transfer data and to evoke or call up the web service functionality:

One approach to invoking web service object would be to call vendor-specific objects directly from applications. That approach requires the application to know details about each vendor specific object, and ties the application to a particular web services vendor. Therefore a generic web services object is introduced. Each vendor specific object is mapped to the generic web services object by a definition stored in an XML mapping file. Applications interact with the generic web services object and need not contain hard-coded dependencies on vendor specific objects.

(Ex-1005, ¶[0074].) *Huang* continues that “[t]he WSOBJ class represent each attribute of a vendor-specific Web Service object as a vendor-independent generic object.” (*Id.*, ¶[0075].)

121. Figure 1C of *Huang* depicts a “lookup” interaction (Ex-1005, ¶[0068]) and shows the necessity of these generic objects (*symbolic names*) as a bridge between the **application component 170** and the “Opportunity” web service object 177 and attribute 179 in **Salesforce web service 147** for utilizing a web service object:



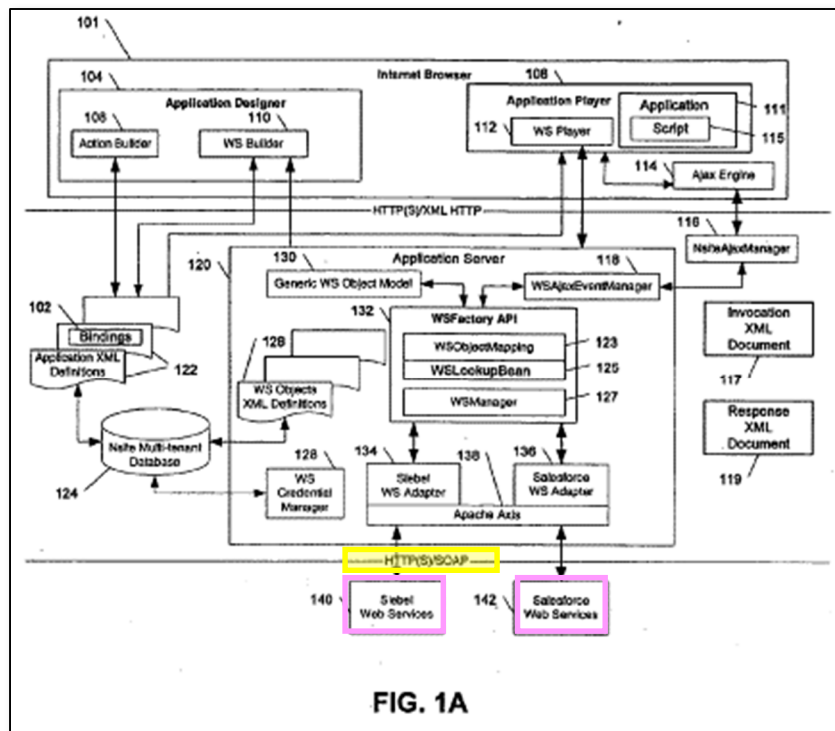
(Ex-1005, Figs. 1C.)

As can be seen, the value 182 (“BigCorp”) of the application component 170 (“OppNameTextArea”) is connected to attribute 179 (“OppName”) of the

“Opportunity” object 177 in Salesforce web service 147 via the generic objects (*symbolic names*).

122. In my opinion, a POSITA would have understood from the above disclosure of *Huang* that these *symbolic names* are necessary and *required* for *Huang*’s system to call or *evoke web services*.

123. *Huang* also discloses that web service 156 is *obtainable over a network*. *Huang* explains that its “server 120 communicates with the web services via a network protocol.” (Ex-1005, ¶[0043].) Figure 1A of *Huang* also shows that **web services** are accessed via a network protocol, in this case “HTTP(S)/SOAP.”



(Ex-1005, Fig. 1A.)

124. Further, a POSITA would have understood that a web service is obtainable over the World Wide Web, which is a network.

125. To the extent *Huang* does not explicitly disclose utilizing *a registry* as recited in element 1.a, a POSITA would have found a *computer memory storing a registry of: a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and b) the address of the web service* obvious over *Huang* in view of *Shenfield*.

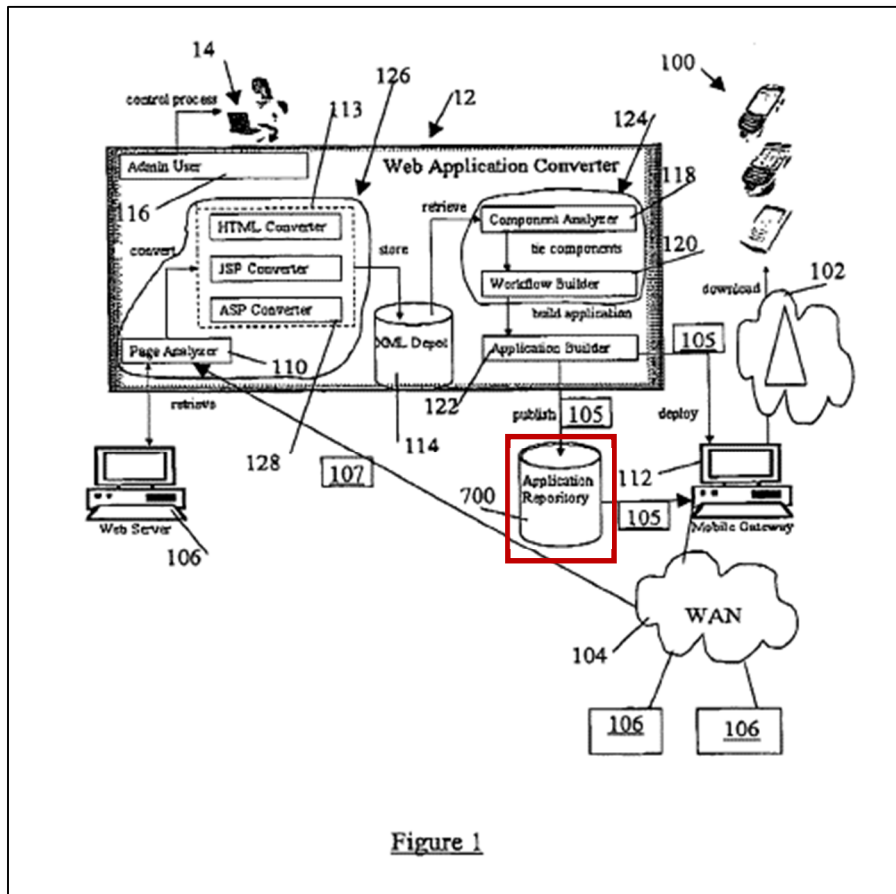
126. As mentioned above, *Shenfield* discloses a web service *registry*. (*Supra* Section X.B.1.b.) Indeed, *Shenfield* discloses that web services registries, such as the UDDI registry, are used to obtain information regarding web service interfaces and therefore can be used to provide the information required to bind user interface components with the backend logic offered by web services. (Ex-1007, ¶¶[0024]-[0025], [0042].)

127. *Shenfield* explains that its components may interact with web services through communication protocols to “expos[e] relevant business logic (methods) of the web services 106 to the component application(s) 105 provisioned on the device 100.” (Ex-1007, ¶[0024].) *Shenfield* also explains that “web service 106 can be defined as a software service, which can implement an interface such as expressed

using Web Services Description Language (WSDL) registered in a Universal Discovery Description and Integration (UDDI) services registry, and can communicate through messages with client devices 100 by being exposed over the network 104 through an appropriate protocol such as but not limited to the Simple Object Access Protocol (SOAP).” (*Id.*, ¶[0025].) *Shenfield* further explains that SOAP defines the format for messages that are used to call up a web service, stating that the “SOAP request message can contain a callable function, and the parameters to pass to the function, which is sent (according to the message and data format described in the components 400, 404 of the component application 105) from the client device 100, and the service 106 then returns the response message (also according to the expected message and data format described in components 400, 404) with the results of the executed function.” (*Id.*; *see also id.*, ¶[0042] (describing message components and how message components can “uniquely represent (and map to) WSDL messages”).) These parameters, data formats, and expected messages are all generally described in the WSDL registered in the UDDI registry. (Ex-1019, ¶[0006]; Ex-1018, 88-90; Ex-1020.)

128. *Shenfield* also identifies **metadata or application repository 700** as a location that can store a web service registry. (Ex-1007, ¶[0044] (“Web Service 106 registry in a **metadata repository 700** (see FIG. 1) as a bundle of platform-neutral data 400, message 404, workflow 406 component descriptors with a set of platform-

specific presentation component 402 descriptors for various pre-defined client runtimes (i.e. specific runtime environments 206—see FIG. 2).”).)



(Ex-1007, Fig. 1.)

129. A POSITA would have understood from *Shenfield*'s disclosures that a web service registry provides the requisite information to interface with a web service. As discussed above, well-known registries, such as the UDDI registry explicitly referenced in *Shenfield*, provide information for, among other things, the data formats, inputs, outputs, and communication structure to access a web service. (Ex-1018, 89.) A POSITA also would have further understood that the registries

disclosed in *Shenfield* provide the information necessary for *Huang*'s system to identify vendor-specific objects and the generic object classes that correspond to such objects. Indeed, *Huang* explains that "web service interface descriptions are typically provided by the web service vendor in a format such as WSDL" (Ex-1005, ¶[0045]), which is the same description that *Shenfield* identifies is available in the UDDI registry (Ex-1007, ¶[0025]).

130. In my opinion, a POSITA would have found it obvious to include a registry containing *Huang*'s *symbolic names*. A POSITA would have recognized and found it obvious (and in fact, likely necessary) that a web services registry includes all the necessary information to interface with a web service and, in the context of *Huang*'s system, the symbolic names would have been included as a part of such necessary information. I refer to my discussion below for further discussion of why a POSITA would have found combining *Shenfield*'s registry with *Huang* obvious. (*Infra* Section X.B.1.b(v).)

- (iii) "where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and"

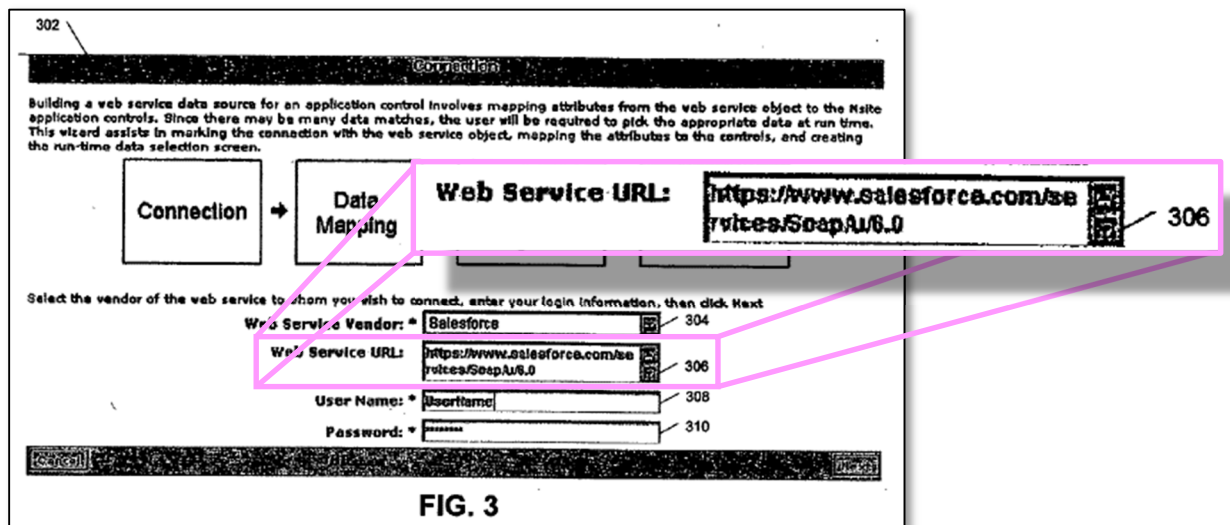
131. In my opinion, *Huang* discloses that its *symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service*. *Huang*'s names for its generic web object attributes, such as "Opportunity," "OpportunityName," and "OpportunityId." (Ex-1005,

¶[0047]), are alphanumeric character strings that do not contain any address or resource location information and, accordingly, do not include a persistent address or a pointer to an output value accessible to the web service.

- (iv) “[where the registry includes:] b) the address of the web service”

132. In my opinion, a POSITA would have found including the address of the web service in the registry that would be implemented in *Huang’s* system as modified by *Shenfield* obvious.

133. *Huang* discloses that an address of a web service is used to locate the web service. Specifically, *Huang’s* system calls for a user to input a “URL 306 of the web service” (Ex-1005, ¶[0105]), and the system creates its generic objects “based on [the] URL supplied” by the user (*id.*, ¶[0077].) I reproduce Figure 3 below, showing the *address* of **web service 156** that is used to locate a particular web service provided by a particular vendor (Ex-1005, ¶[0077]):



(Ex-1005, Fig. 3.)

Huang explains that “[t]he URL refers to a web service provided by a particular vendor” and, once this URL is inputted, the system “retrieves objects from the web service via a communication protocol such as SOAP (Simple Object Access Protocol) over HTTP (HyperText Transfer Protocol).” (*Id.*)

134. Similarly, *Shenfield* also discloses that an address of a web service is used to locate the web service through its reference of an “interface such as expressed using Web Services Description Language (WSDL) registered in a Universal Discovery Description and Integration (UDDI) services registry.” (Ex-1007, ¶[0025].) A POSITA would have understood that the WSDL description of a web service would include the web address of where to access the web service. (*E.g.*, Ex-1018, 89 (listing “the network address” as a necessary piece of information”); Ex-1019, ¶[0006] (UDDI may provide “the location of the web service (e.g., its URI specified by a specific network destination address or name)”)).

135. In my opinion, a POSITA would have found including the address of the web service in the registry of the modified *Huang-Shenfield* system obvious because a POSITA would have been motivated to—and, in fact, recognized that it would have been necessary to—include this information in the registry and would have had a reasonable expectation of success because this is standard information included in registries and would not be expected to interfere with the operation of a

system. I also refer to my analysis below as to why a POSITA would have found modifying *Huang*'s system to include a registry obvious. (*Infra* Section X.B.1.b(v).)

(v) A POSITA Would Have Been Motivated to Modify *Huang* with *Shenfield*'s Registry with a Reasonable Expectation of Success

136. It is my opinion that *Shenfield* and *Huang* are in the same field and share objectives such that a POSITA would have been motivated to look to *Shenfield* when seeking to implement the system of *Huang*. *Shenfield*, like *Huang*, relates to accessing network-accessible web services. (Ex-1007, ¶[0001] (“communication of services over a network”); Ex-1005, ¶[0002] (“This invention relates to ... web services.”).) Both also address the problem of cross-functionality across multiple platforms. (See Ex-1007, ¶[0003] (stating need for programs “that can be run on client devices having a wide variety of runtime environments”), ¶[0005] (same); Ex-1005, Abstract (“allow use of web services across organizations”), ¶[0003] (“integrating [] various software applications with one [an]other”).)

137. In my opinion, a POSITA would have found *Shenfield*'s disclosures of utilizing a registry of web services to be relevant to and compatible with implementing *Huang*'s system. For example, *Huang* discloses generating mapping files (or adaptors) that relate its “Generic WS Object Model 130” to vendor-specific objects by “access[ing] a WSDL (Web Service Definition Language) description for each vendor” via, for example, SOAP. (Ex-1005, ¶¶[0008], [0044]-[0045], [0077].)

Shenfield discloses accessing the same information “in a Universal Discovery Description and Integration (UDDI) services registry” and utilizing the same protocol “Simple Object Access Protocol (SOAP)” to allow “client devices 100” to communicate with the web service. (Ex-1007, ¶[0025].) It is my opinion that a POSITA would have understood that both *Huang* and *Shenfield* contemplate utilizing the same binding information (WSDL) and interfacing with web services using the same protocol (SOAP) and, therefore, *Huang*’s system could be readily modified to utilize a web services registry, such as the UDDI registry, as disclosed in *Shenfield*.

138. A POSITA would have also found it obvious to store the registry in the modified *Huang-Shenfield* system in *Huang*’s database 124. *Shenfield* discloses that a “web services 106 registry” can be stored in “application” or “metadata repository 700” of *Shenfield*’s system. (Ex-1007, ¶[0044] & Fig. 1.) A POSITA would have understood that *Huang*’s database 124 and *Shenfield*’s metadata repository 700 are analogous computer memories because, for example, both store similar types of data, both receive applications generated by the system and deliver them to end-users, and both are stand-alone components in communication with the system and end users over a network. (Ex-1005, ¶[0024] & Fig. 1A (XML definitions of *Huang*’s applications provided to browser-based players over a network); Ex-1007, ¶[0044]

& Fig. 1 (*Shenfield*'s registry and bundles of component applications provided to devices over a network).)

139. In my opinion, a POSITA would have understood that the resulting system of *Huang*, as modified by *Shenfield*, would include *Huang*'s web components, such as the "Opportunity" object in the "Generic WS Object Model 130," which I discuss in Section X.B.1.b(ii); the symbolic names of the attributes (which are the inputs and outputs) of a web service, such as *Huang*'s "OpportunityName" and "OpportunityId" attributes for the "Opportunity" web object, which I also discuss in Section X.B.1.b(ii); and the addresses of the web services. As discussed above, *Huang* explains that its symbolic names for web service objects (*web components*), and their attributes (*inputs and outputs of web services*) are necessary intermediaries to interact with web services. (*Supra* Section X.B.1.b(ii).) Further, a POSITA would also have recognized that the addresses of web services, such as their URLs, would be required to access the specified web service. (*Supra* Section X.B.1.b(iv); Ex-1018, 89 (listing "the network address" as a necessary piece of information").)

140. A POSITA would have understood that these components are necessary for the applications of *Huang*'s system to interact with web services and therefore would be necessary pieces of information that should be included within the web service registry of the *Huang-Shenfield* system. Indeed, *Huang* confirms that such

information is necessary. For example, *Huang*'s exemplary mapping and binding files include the symbolic names of the web object and attributes and the web address of a web service. (Ex-1005, ¶¶[0136], [0139].):

APPENDIX D

```
<WebServiceBinding vendormname="Salesforce"
  objectname="Opportunity"
  username="" wsurl="https://www.salesforce.com/services/Soap/u/6.0 ">
  <WSAttributeBindingList>
    <WSAttributeMapping formentity="OppNameTextArea"
  wsattribute="OppName"/>
    <WSAttributeMapping formentity="DescriptionTextArea"
  wsattribute="Description"/>
    <WSAttributeMapping formentity="StageTexArea"
  wsattribute="StageName"/>
    <WSAttributeMapping formentity="ClosedCheckbox"
  wsattribute="IsClosed"/>
  </WSAttributeBindingList>
```

(Ex-1005, ¶[0139]; *see also id.*, ¶[0059].)

141. In another example, *Huang*'s system requires a user to input a URL and credentials to access a web service. (Ex-1005, ¶[0105], Fig. 3.) *Huang* further discloses that its system "accesses a WSDL (Web Service Definition Language) description for each vendor" to generate the generic objects and the mapping of the generic objects to vendor-specific objects. (*Id.*, ¶¶[0008], [0045].) *Huang* further discloses that its "WSManager 127 may be invoked by the Designer 104, e.g., to get a list of web service objects," among other things. (Ex-1005, ¶[0049].) Thus, a POSITA would have understood that *Huang* contemplates accessing a WSDL

document and obtaining web service, web component, and web service input and output information each time a user identified a relevant web service to generate its generic objects, such as the “Opportunity” web service object.

142. In my opinion, a POSITA would have understood that the registry of the *Huang-Shenfield* system must contain all necessary information, such as the mapping, binding, and address information disclosed in *Huang*, to access a web service because a POSITA would have understood that the registry of the *Huang-Shenfield* system must provide all necessary information to access a web service to serve its purpose as a data store of binding information. Put differently, the registry would be unable to serve this function if it were missing critical information. Accordingly, a POSITA would have found including the symbolic names and address of a web service in a web services registry of the modified *Huang-Shenfield* system both obvious and necessary.

143. In my opinion, a POSITA would have been motivated to implement a registry of web services, their web components, and the symbolic names corresponding to the inputs and outputs of the web services that relate to each web component because utilizing a registry would improve system performance and minimize the burdens on system resources. For example, utilizing a registry containing the necessary information to access a web service would eliminate the need to analyze each web service for its web components and the associated inputs

and outputs every time a user attempts to locate a web service because the registry would already contain such information. (Ex-1005, ¶[0049] (“The WSManger 127 may be invoked by the Designer 104, e.g., to get a list of web service objects.”).) Further, a registry incorporating the address of a web service would eliminate the need for a user to have preexisting knowledge of web service addresses. Instead, a registry would enable a search functionality that could simplify the identification of proper vendors that provide web services appropriate for certain criteria. Further, because the web addresses would also be included in the registry, *Huang’s* system could maintain its original functionality of using a web address as the relevant identifier for a known web service. (Ex-1005, ¶¶[0077], [0105], Fig. 3.) A POSITA would have been motivated to implement a registry to *Huang’s* system to realize these benefits in data-processing efficiency and record-keeping relating to *Huang’s* web services.

144. It is also my opinion that a POSITA would have been motivated to implement a registry in the system of *Huang* because it would have been a straightforward and conventional way to implement a system that interfaces with or accesses a web service. For example, POSITAs were aware that symbolic names were used for accessing various types of data and services. (Ex-1028, Abstract, 1:10-11, 1:39-40; Ex-1029, ¶¶[0005], [0032], [0035].) Therefore, a POSITA would

have found implementing a registry of web components an obvious option that could simplify and streamline data processing.

145. It is also my opinion that a POSITA would have been motivated to store the registry in *Huang*'s database 124 to leverage already-existing data stores in *Huang*'s system. Indeed, *Shenfield*'s disclosures of storing a web services registry in application repository 700 (Ex-1007, ¶[0044]) would have directed a POSITA to identify corresponding data stores in *Huang*, with database 124 being a primary candidate that had the same relevant characteristics. At a minimum, database 124 is a recognized computer memory that could store a registry and would have been one of a finite number of options that could be utilized by a POSITA. Accordingly, a POSITA would have found storing registry in database 124 obvious to try.

146. It is also my opinion that a POSITA would have reasonably expected modifying *Huang* to implement a web component registry containing the symbolic names of web components and their attributes, as well as the addresses of web services, in database 124 to succeed. As discussed above, web service registries were well known (*e.g.*, Ex-1019, ¶¶[0005]-[0007], Figs. 1-2; Ex-1007, ¶¶[0025], [0044]; *see also* Ex-1004, 1068-69 (explaining WSDL provides symbolic names for web services)), such as the UDDI registry, and implementing the use of a web component registry would have been easy in view of the knowledge of a POSITA and unlikely to interfere with system function because the registry would provide

the same information as what would be provided by *Huang*'s system analyzing each vendor's WSDL on a per request basis. Relatedly, a POSITA would have understood that symbolic names and addresses were commonly used in registries (Ex-1019, ¶¶[0005]-[0007], Figs. 1-2; *see also* Ex-1004, 1068-69 (explaining WSDL provides symbolic names for web services)) and, therefore, a POSITA would have understood how to implement registries and would not expect the inclusion of a registry to interfere with the function of the system. Also, *Huang*'s system and *Shenfield*'s system both utilize the same type of information and communication protocol to obtain information regarding how to interface with a web component—specifically, both utilize information in a WSDL description of the web service and access this information using SOAP communications. (*Compare* Ex-1005, ¶¶[0008], [0044]-[0045], [0077] *with* Ex-1007, ¶[0025].) Accordingly, modifying *Huang*'s system to utilize the registry disclosed in *Shenfield* would have been technically compatible and easy to implement. Further, storing the registry in database 124 would have been easy to implement because database 124 is sufficiently analogous to application repository 700 of *Shenfield*'s system in terms of relevant characteristics such that there would have been no reason for a POSITA to believe that this would meaningfully impact the functioning of the system. It is my opinion that, at a minimum, this modification would have been a simple substitution of known elements to achieve predictable results.

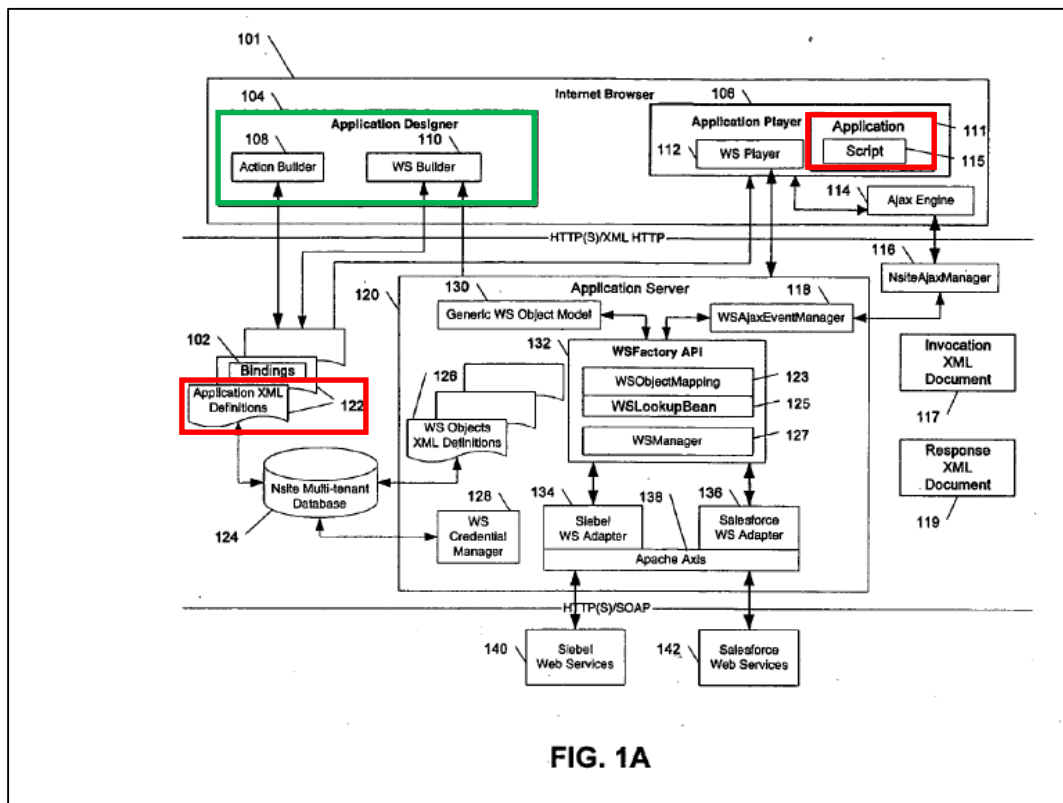
- c. [1.b] “an authoring tool configured to: define a user interface (UI) object for presentation on the display, where said UI object corresponds to the web component included in said registry selected from the group consisting of an input of the web service and an output of the web service,”

147. It is my opinion that *Huang* in view of *Shenfield* discloses or suggests element 1b. I understand that the district courts adopted the construction of “authoring tool” and “authoring tool configured to” to mean “a system, with a graphical interface, for generating code to display content on a device screen,” which is a construction to which Express Mobile agreed. (Ex-1011, 10.) Based on the analysis below, it is my opinion that *Huang* in view of *Shenfield* renders element 1.b obvious under the construction the district courts adopted. I address portions of element 1.b separately for purposes of clarity.

- (i) “an authoring tool configured to: define a user interface (UI) object for presentation on the display,”

148. In my opinion, a POSITA would have understood that *Huang* discloses *an authoring tool for defining a user interface (UI) object for presentation on the display*. For example, *Huang* discloses “defining an application component in [*Huang*’s] application designer.” (Ex-1005, ¶[0103].) *Huang* explains that its **designer 104** (*authoring tool*) is a part of its *system* and “is a computer program that provides a user interface which allows a user to create the **application 111**,” and that these **applications** are stored as “**XML definitions 122** in a database 124” until they

are “loaded into Player 106 as the **application 111**” when executed. (Ex-1005, ¶¶[0040], [0042].)

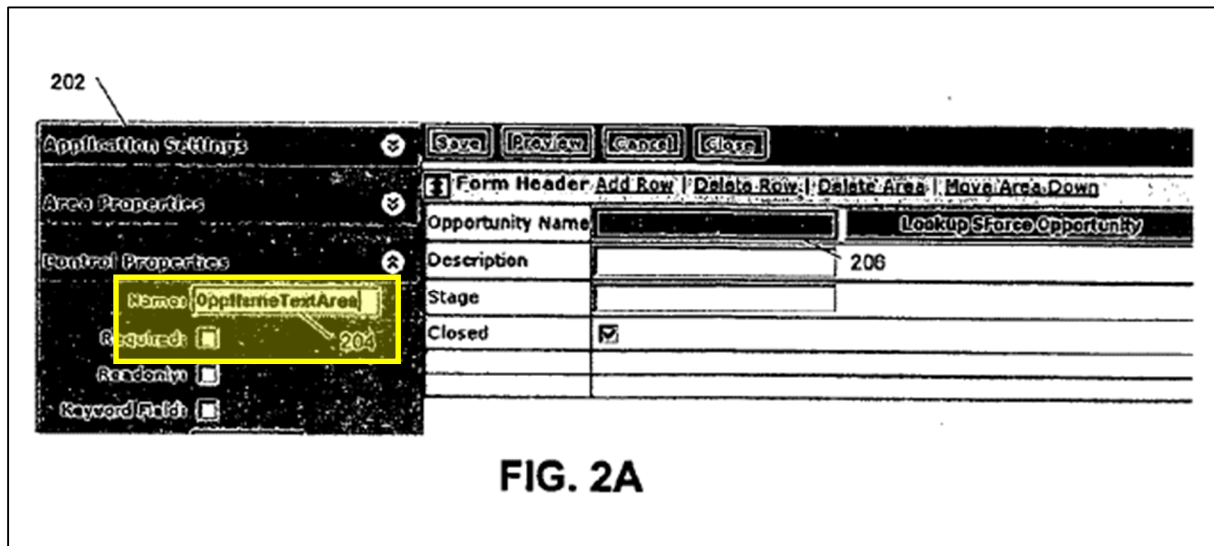


(Ex-1005, Fig. 1A.)

Thus, a POSITA would have understood that *Huang* describes **Designer 104** as a software tool that allows a user to generate code in the form of *Huang*’s applications, such as **application 111** and **application XML definitions 122**, which are code (XML) used to display content on the display of a device.

149. It is also my opinion that *Huang*’s designer 104 allows a user to *define user interface (UI) objects for presentation on the display* through a graphical user interface. *Huang* explains that designer 104 “provides a user interface for *defining*

web services interactions in browser-based applications.” (Ex-1005, ¶[0054]; *see also id.*, ¶[0040].) For example, *Huang* explains that Figure 2A “is an illustrative drawing of *defining* an **application component** in an application Designer. ... The **application component 206** is a text input component and has an associated name, 204, OppNameTextArea.” (*Id.*, ¶[0103].)



(Ex-1005, Fig. 2A.)

150. *Huang* also discloses that that these application components allow for users to interact with, for example, bound web services by inputting data or display information, such as values retrieved from a web service. (*E.g.*, Ex-1005, ¶¶[0103] (“The application component 206 is a text input component and has an associated name, 204, OppNameTextArea.”), [0118] (“The application 1102 includes a Lookup SForce Contact button 1104, which was defined using the application Designer A user of the application 1102 can press the Lookup SForce Contact button 1104 to

display a Selection Lookup Table for retrieving values for the application components.”); *see also id.*, ¶¶[0067] (describing “application component[s]” allowing for, e.g., “lookup, insert, modify or delete” interactions), [0070] (describing exemplary “lookup interactions”), [0073] (discussing “button” components “configured to invoke a web service”).) *Nachnani*, which discloses “[t]he sort of software applications ... [that] may also be used in the system of [*Huang*]” (Ex-1005, ¶[0006]), explains that each component is defined by “properties associated with each component type” and that “the component is displayed” unless set to be hidden (Ex-1006, ¶¶[0042]-[0043]). Accordingly, a POSITA would have understood that *Huang*’s application component is a *user interface (UI) object that is defined and is presented to a user on a display*.

151. A POSITA would have also understood that *Huang*’s designer is a graphical user interface. The designer allows a user to generate code in the form of *Huang*’s applications and does so through interactive graphical elements presented to the user, such as what is shown in Figure 2A. As shown in Figure 2A, *Huang*’s designer displays interactive graphical elements that allows a user to generate code that defines the behavior and functionality of, for example, the “OppNameTextArea” text input field and butts, which are elements to be displayed to a user to allow for interaction, such as inputting text or clicking.

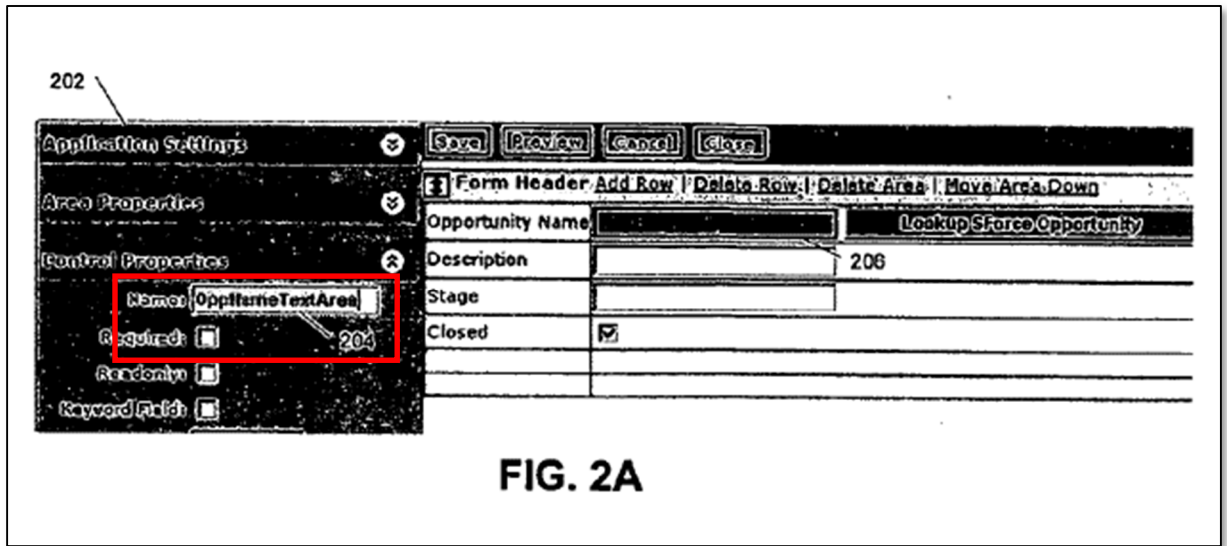


FIG. 2A

(Ex-1005, Fig. 2A.)

- (ii) “where said UI object corresponds to the web component included in said registry selected from the group consisting of an input of the web service and an output of the web service,”

152. In my opinion, a POSITA would have found an application component, which is the *UI object*, corresponding to the web component included in the registry selected from the group consisting of an input of the web service and an output of the web service obvious over *Huang* in view of *Shenfield*. As previously discussed, a POSITA would have found modifying *Huang*’s system in view of *Shenfield* to utilize a registry including symbolic names for *Huang*’s attributes, which are identifiers for the *inputs and outputs of the web service*, such as “OpportunityName” and “OpportunityId” attributes) obvious. (*Supra* Section X.B.1.b.)

153. It is further my opinion that *Huang* discloses that the *UI object corresponds to the web component*. I note again that in the modified *Huang*-

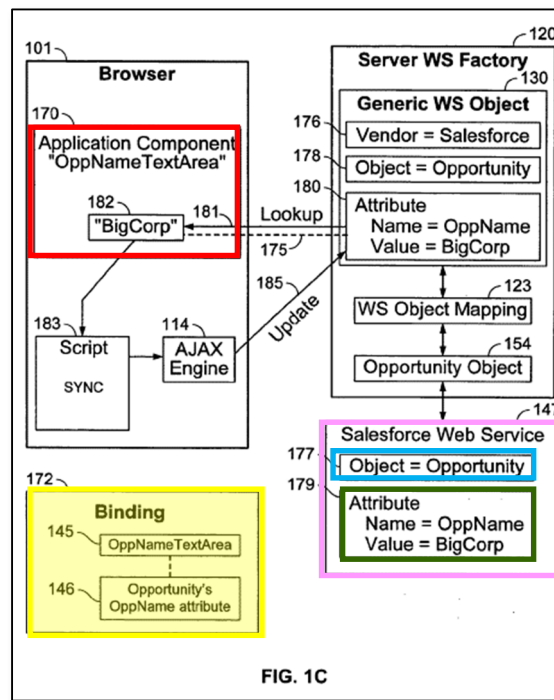
Shenfield system, this web component would have been included in the web services registry.

154. For example, *Huang* discloses the “OppNameTextArea” application component is bound to the “OppName” attribute of the “Opportunity” web service object. (*E.g.*, Ex-1005, ¶[0060] (“[T]he OppNameTextArea application component is bound to the OppName web service attribute.”).) The “OppNameTextArea” application component is an example of a *UI object* because *Huang* describes this as “a text input component” with which a user can interact. (*Id.*, Ex-1005, ¶[0103].)

155. The “OppNameTextArea” *UI object* corresponds to the “OppName” attribute of the “Opportunity” web object, which is a *web component*. Specifically, *Huang* explains that “[t]he binding 193 ... specifies that a particular web service attribute 155 can be set to the value of the application component 170, and, conversely, that the value of the application component 170 can be set to the value of the attribute 155 ... as part of a web service interaction such as a lookup, insert, update, delete, or user-defined interaction.” (*Id.*, ¶[0056].) Put differently, the binding allows the application component to set or input its value to the corresponding attribute of the web service object or the application component to get or receive the output value of the corresponding attribute.

156. For example, the *correspondence* between the “OppNameTextArea” text input **application component 170** (*UI object*) and the “OppName **attribute 179**

(input and output of a web service) of the “Opportunity” **web object 177** (web component) from the Salesforce **web service 147** can be seen in Figure 1C below, which shows a “lookup” operation where, through binding 172 between “OppNameTextArea” and “Opportunity’s OppName attribute,” the value of “OppNameTextArea” is set to the value of the “OppName” attribute.



From these disclosures, a POSITA would have understood that *Huang* discloses that the *UI objects correspond to a web component selected from the group consisting of an input or output of a web service*.

157. Because the combined *Huang-Shenfield* system would utilize a *registry* that includes the *symbolic names* of the web service object (*web component*) and its attributes (*inputs and outputs of a web service*), the combined *Huang-Shenfield*

system discloses and renders obvious that the application component (*UI object*) corresponds to the web component included in the registry selected from the group consisting of an input and an output of the web service.

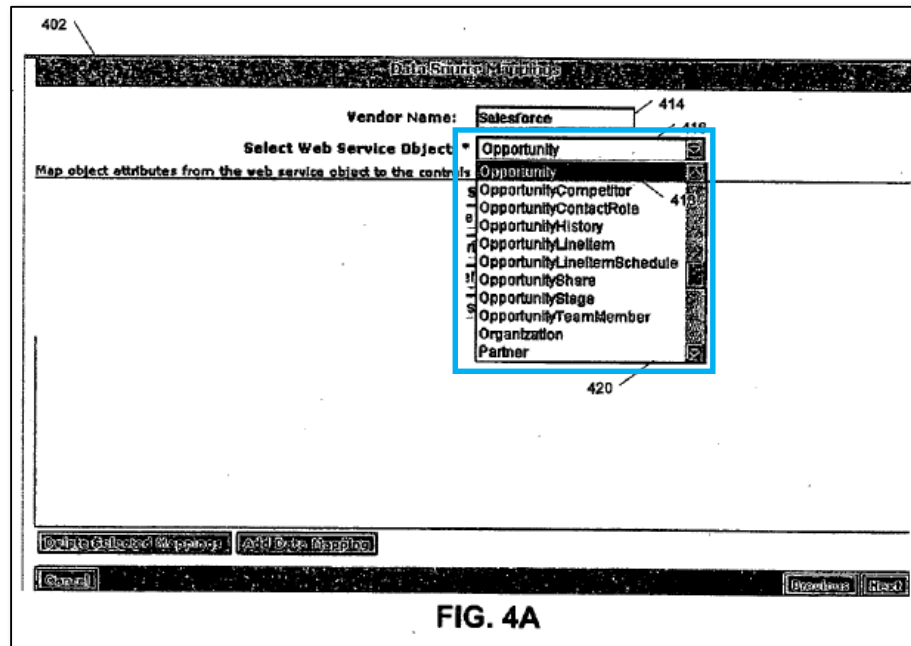
d. [1.c] “[authoring tool configured to:] access said computer memory to select the symbolic name corresponding to the web component of the defined UI object,”

158. In my opinion, *Huang* in view of *Shenfield* renders element 1.c obvious. *Huang* discloses that a user, through *Huang*’s designer (*authoring tool*), selects a symbolic name of a web object (*web component*) and an attribute of that web object (*input and output of the web service*) that will be bound to, and thus correspond to, the application component (*the defined UI object*). (*Supra* Section X.B.1.c.) Further, a POSITA would have understood that the modified *Huang-Shenfield* system would access the registry stored in *Huang*’s database 124 (*computer memory*) to access the symbolic names of the relevant web service to allow for selecting the symbolic name corresponding to the web component of the defined UI object.

159. *Huang* discloses selecting the symbolic name corresponding to the web component of the defined UI object through its *authoring tool* in Figures 4A-4C.

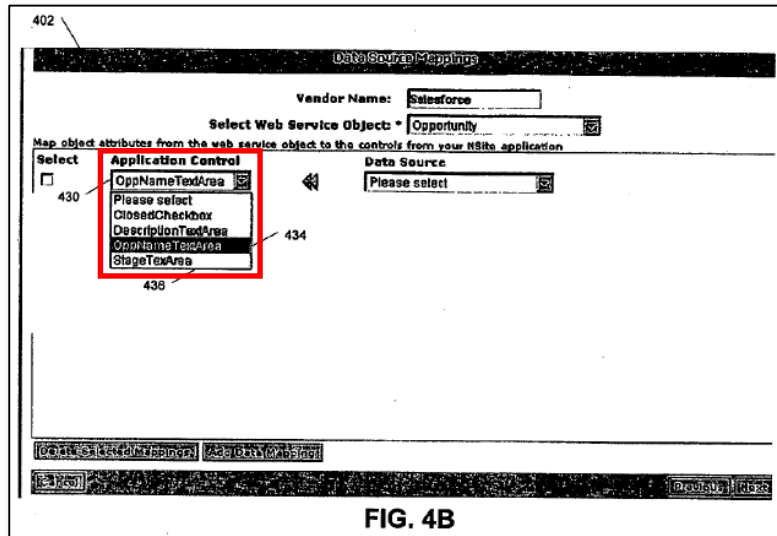
160. In Figure 4A, *Huang* depicts a user of designer 104 “choos[ing] a selected web service object 416.” (Ex-1005, ¶[0106].) “[W]eb service object 416” is a **web component** for the reasons discussed above. (*Supra* Section X.B.1.b(ii).)

The user selects the symbolic names representing **web components** from “selection menu 420” (Ex-1005, ¶[0106]):



(Ex-1005, Fig. 4A.)

161. The user then “choose[s] an **application component 420** from the selector 436.” (Ex-1005, ¶[0107].) In this example, the options include the “OppNameTextArea” text input user interface (*defined UI object*) previously discussed.



(Ex-1005, Fig. 4B.)

162. *Huang* explains that “[a] binding is created by selecting an **application component** and a **web service attribute** using the application component selector 430 and the web service attribute selector 440, respectively.” (Ex-1005, ¶[0108].) Figure 4C shows the process of selecting the **web service attribute** from a list of web service attributes shown in web service attribute selector 440. In this example, the **attribute** having the symbolic name “Name” has been selected.

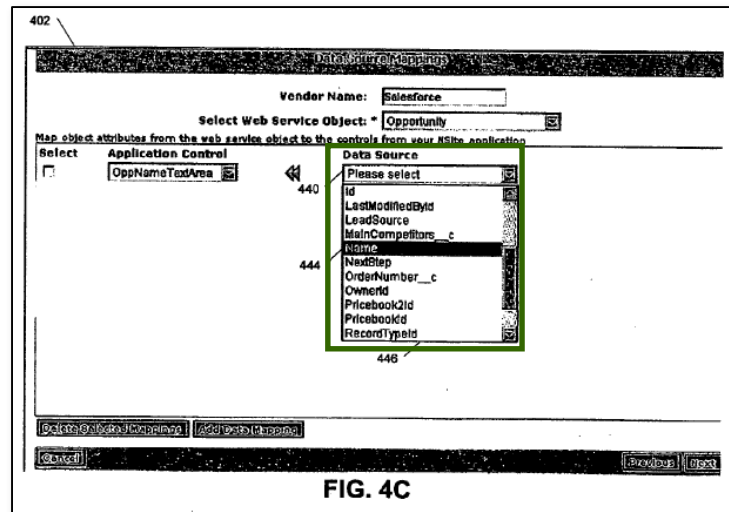
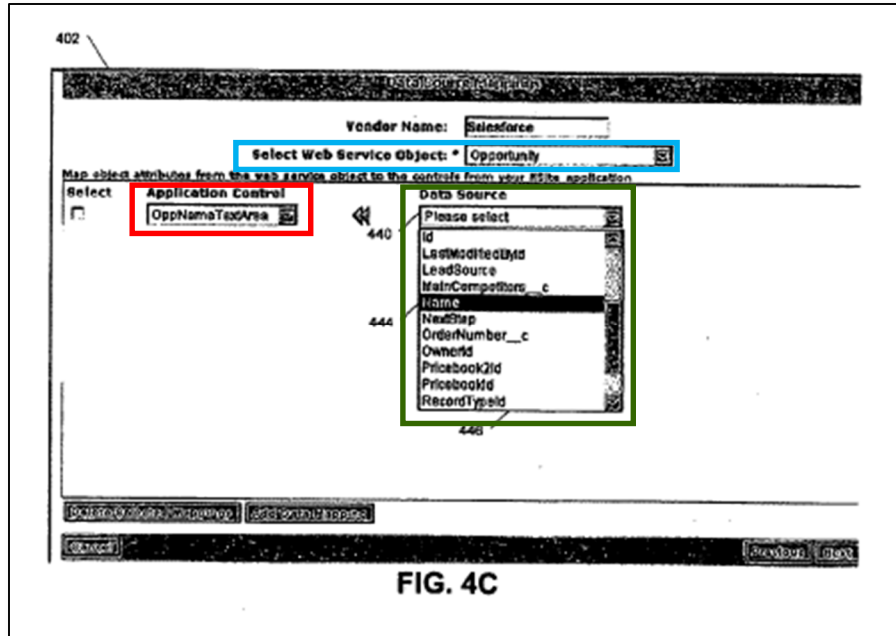


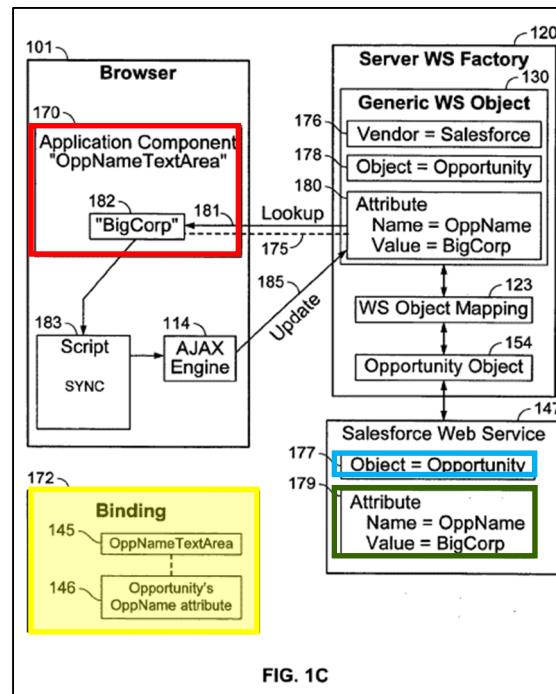
FIG. 4C

(Ex-1005, Fig. 4C.)

163. Thus, in this example, the user of the designer *selected* the “Name” (symbolic name) **attribute** (input and output of a web service) from the “Opportunity” **web service object** (web component), which is bound to and thus corresponds to the “OppNameTextArea” **application component** (defined UI object), as can be seen in Figures 4C and 1C below.



(Ex-1005, Fig. 4C.)



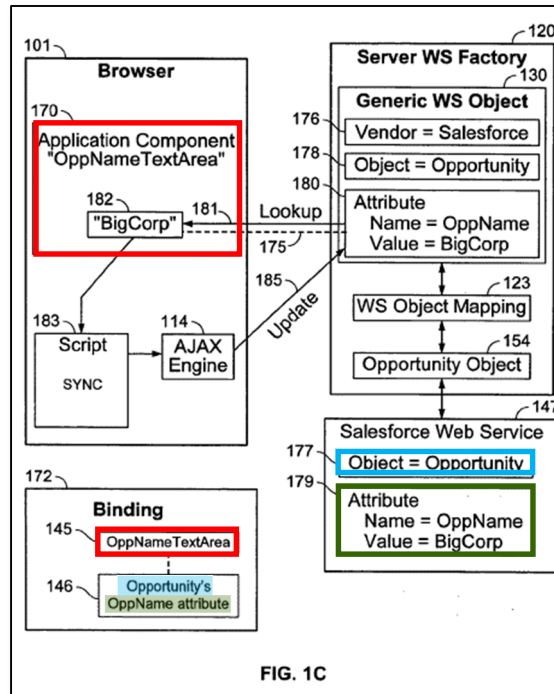
(Ex-1005, Fig. 1C.)

164. Based on the foregoing, it is my opinion that a POSITA would have understood that *Huang* discloses *selecting a symbolic name from said web component corresponding to the defined UI object*.

165. As previously discussed, a POSITA would have found it obvious to modify *Huang*'s system to use the registry disclosed in *Shenfield* and store the registry in *Huang*'s database 124. (*Supra* Section X.B.1.b(i).) I refer to and incorporate by reference by discussion above as if set forth herein. A POSITA would have understood that *Huang*'s designer, which is the *authoring tool*, would access the *registry* stored in the *computer memory* for a list of web objects and attributes in the modified *Huang-Shenfield* system rather than access each vendor web service to identify web objects and attributes.

e. [1.d] “[authoring tool configured to:] associate the selected symbolic name with the defined UI object,”

166. It is my opinion that *Huang* discloses element 1.d. *Huang* discloses that its designer 104, which is the *authoring tool*, enables a user to generate bindings that link (and thus, *associate*) application components (*defined UI object*) with the *symbolic names* of web attributes (*inputs and outputs of a web service*) of a web object (*web component*). The establishment of a binding *associates* an **application component** (*UI object*) and the *symbolic name* of an **attribute** (*input and output of a web service*) of a **web object** (*web component*) to allow for the flow of data between the web service and the UI object, as can be seen in Figure 1C below.



(Ex-1005, Fig. 1C.)

- f. [1.e] “[authoring tool configured to:] produce an Application including the selected symbolic name of the defined UI object, where said Application is a device-independent code”

167. In my opinion, *Huang* discloses element 1.e. As discussed above, the district courts construed “Application” to mean “device-independent code containing instructions for a device and which is separate from the Player” and “device-independent code” to mean “code that is not specific to the operating system, programming language, or platform of a device.” (*Supra* Section VII.) As discussed below, *Huang* discloses element 1.e under the district courts’ constructions of these terms.

168. *Huang* discloses that its designer *produces an application* and that the *application is separate from a player and is a device-independent code*. For example, Figure 1A depicts the **Designer 104** generating **application 111** (as **XML definition 122**), which is separate from and ultimately executed in **player 106**:

Huang discloses that the designer creates an application and that the above-discussed associations between application components and web service attributes are “added” to the application. (Ex-1005, ¶¶[0040], [0108].)

wizard” that “creates the binding or mapping of meta-information to generate composite or other software applications.” (*Id.*, ¶[0008].) A POSITA would have understood that *Huang*’s references to “creat[ing]” and “generating” applications to mean that *Huang*’s system produces an application.

170. In my opinion, a POSITA would have understood that *Huang* discloses that its application, such as “application 111,” is written in device-independent code. *Huang* identifies that its applications “are stored in Extensible Markup Language (SML) format as XML definitions 122 in a database 124.” (Ex-1005, ¶[0042].)

171. *Huang* also incorporates by reference *Nachnani*, which provides additional details regarding the XML code. (Ex-1005, ¶¶[0006], [0042].) *Nachnani* explains that “[a]pplication definitions are specified using the Extensible Markup Language (XML).” (Ex-1006, ¶[0045].) *Nachnani* contains numerous additional disclosures that identify the application as a definition written in XML format. (*Id.*, ¶¶[0056] (“FIGS. 4a and 4b illustrate a schema describing a format for an application definition according to one embodiment of the invention. The schema may be, for example, an XML Schema describing the format of an XML document that represents an application. An application is stored, e.g. in a database, as an XML document that conforms to a Form schema 410.”), [0063] (“The application definition 540 is a text document in Extensible Markup Language (SML) format and includes nested elements definitions.”), [0097] (“In one embodiment, the Designer

client 803 saves the application as an application definition 808, which is a description of the application in a format such as XML.”), [0103] (“an application definition in the form of an XML document”), [0104] (“uses an XML application definition”), [0106] (“creating an XML application definition”).) *Nachnani* further explains that Figure 2B shows “an XML document structure 261 [that] shows how an application 200 ... can be described using XML” and that the elements within the application, such as the “FormComponent element 257” is also defined in XML. (*Id.*, ¶¶[0048], [0052].)

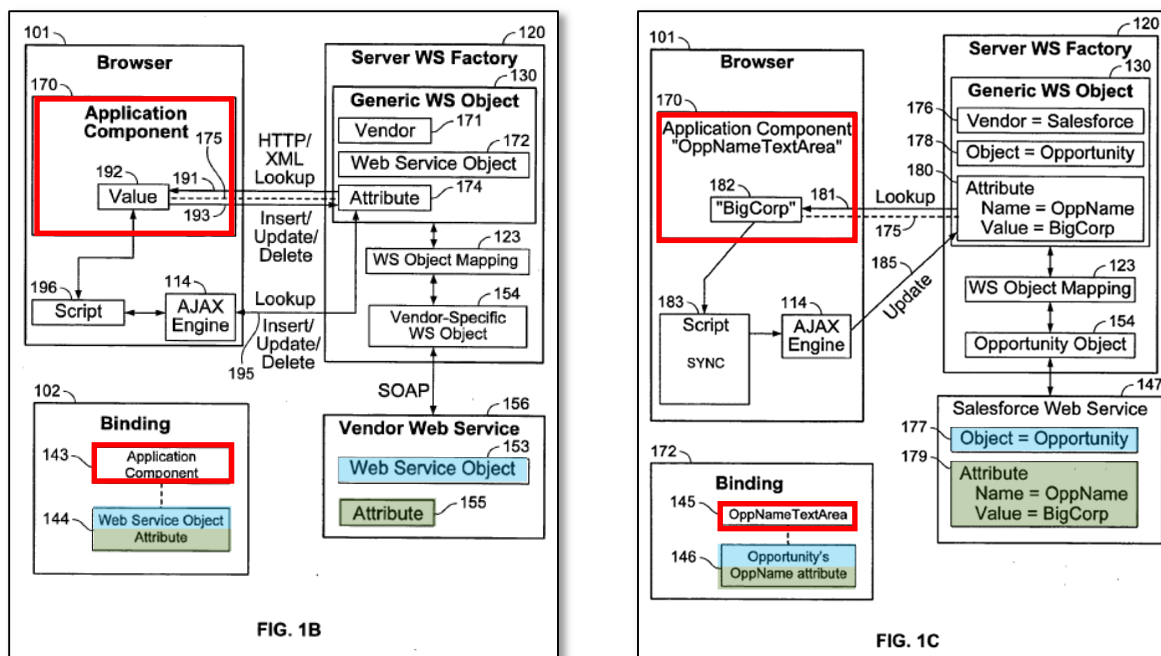
172. A POSITA would have understood that *Huang*’s disclosures (including the disclosures of *Nachnani*) that its applications are XML representations disclose that the application is written in device-independent code under the district courts’ construction that “device-independent code” is “code that is not specific to the operating system, programming language, or platform of a device”. (*Supra* Section VII.) A POSITA would have known that XML is a markup language that utilizes metadata tags to specify the content and structure of data, and such XML descriptions are platform independent. (Ex-1026, 4:3-8; *see also id.*, 3:64-4:3 (“[D]ata types 216 are ***platform-independent because they comply with the XML Standard.***”).) Further, a POSITA would have been well aware that XML is an important aspect of web services because it “permits cross-platform communications,” further confirming its platform independence. (*See* Ex-1027, 19.)

173. It is further my opinion that *Huang*'s application 111 provides instructions for a device. As discussed above, *Huang* and *Nachnani*'s applications include application components (*UI objects*) that interact with web services to input or obtain values for display. (*Supra* Sections X.B.1.a, X.B.1.c.) A POSITA would have understood that *Huang*'s applications, which contain the information for display on a device must therefore provide instructions to the device regarding what to display. Indeed, software components dictate the functions of hardware components, such as a display, through the provision of instructions.

174. It is also my opinion that *Huang* discloses that the application contains the selected symbolic name. In describing the process by which the bindings between application components and selected web service attributes are established, *Huang* explains that these bindings are “added to the application by pressing an Add Data Mapping button.” (Ex-1005, ¶[0108].) *Huang* further explains that the “Applications created by the Designer 104 are stored in Extensible Markup Language (XML) format as XML definitions 122” (*Id.*, ¶[0042]) and the XML definitions include “bindings 102 which associate application components with web service object attributes.” (*Id.*)

175. Further, *Huang* explains that its “[a]pplications interact with the generic web services object” and that “each vendor-specific object is mapped to the generic web services object by a definition stored in an XML mapping file.” (Ex-1005,

¶[0074].) The relationship between *Huang*'s applications, the generic web services objects, and vendor-specific objects, can be seen in, for example, Figures 1B and 1C, reproduced below, which show that the **application component** "OppNameTextArea" (*UI object*) is bound to the *symbolic names* of the **attribute** (*input and output*) ("OppName") of a **web object** (*web component*) ("Opportunity"):



(Ex-1005, Figs. 1B-1C.)

176. In my opinion, a POSITA would have understood from *Huang*'s disclosures that the symbolic name associated with defined UI object is included in *Huang*'s application. A POSITA would have understood from *Huang*'s disclosures that its system "add[s] to the application" the bindings between an application component (the UI object) and a web service object attribute and that the XML

definitions of the application “include bindings 102 which associate application components with web service object attributes” as requiring that the application contain the symbolic names of the web service object and attributes. (Ex-1005, ¶¶[0042], [0108].) *Huang*’s disclosures indicate that the code of the application contains the defined relationships between an application component and the generic web services object that maps to a vendor-specific input or output and, therefore, must contain the identity of the generic web services object, which, as discussed above, is represented by symbolic names in *Huang*’s system.

177. Further, a POSITA would have understood that *Huang*’s application must contain the symbolic names of *Huang*’s web service objects and attributes. *Huang*’s disclosure that its application interacts with the generic web services objects would indicate to a POSITA that *Huang*’s application must necessarily contain the relevant identifier (in the context of *Huang*, the symbolic name) of the user-selected generic object attribute to which it must interact.

g. [1.f] “and [authoring tool configured to:] produce a Player, where said Player is a device-dependent code;”

178. It is my opinion that *Huang* in view of *Shenfield* renders element 1.f obvious. The district courts construed “Player” to mean “device-specific code which contains instructions for a device and which is separate from the Application” and that “device-dependent code” means “code that is specific to the operating system, programming language, or platform of a device.” (*Supra* Section VII.)

(i) Huang and Shenfield Discloses Element 1.f

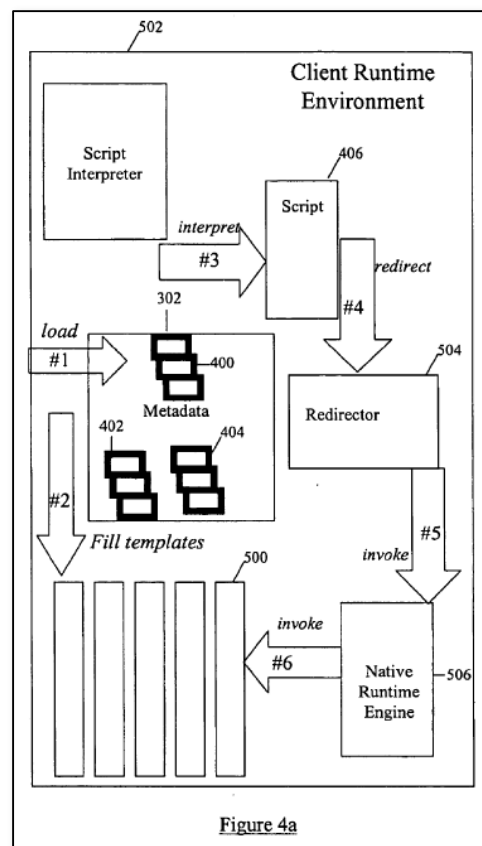
179. In my opinion, *Huang* suggests that its designer 104 generates its “player 106.” As explained in *Huang*, its system is designed such that “Player 106” is necessary to render the application (*e.g.*, Ex-1005, ¶[0042]; Ex-1006, ¶[0040]); therefore, a POSITA would have understood that *Huang*’s system components that generate code, such as designer 104, likewise generates its “Player 106.”

180. *Shenfield* expressly discloses that its system produces component applications, which include components written in platform-independent code that define content and platform-dependent presentation and workflow components. (Ex-1007, ¶[0057] (“the conversion tools 12 are executed on a computer 14 and are used to generate component applications 105”).) A POSITA would have understood that the platform-independent content corresponds to the applications of *Huang* (*e.g.*, Ex-1005, ¶[0040] (“application 111, which can receive values from web services, send values to web services, and invoke other operations provided by web services” when executed in a player) and the *application* recited in the claims of the ’755 patent because these components define data structures and interactions. (Ex-1007, ¶¶[0041] (“the data components 400 define data entities which are used by the component application program 302”), [0042] (“the message components 404 define the format of messages used by the component applications 105 to communicate with external systems such as the web service 106”).) The platform-dependent

presentation and workflow components correspond to *Huang*'s player (*e.g.*, Ex-1005, ¶[0040] (“The Player 106 is a computer program that provides a user interface which executes a browser based application 111.... Player 106 execute[s] in a conventional Internet Browser 1010....”) and the *player* recited in the claims of the ’755 patent because they present the applications to the user and are used to execute the functionality defined by the platform-independent components. (Ex-1007, ¶¶[0043] (“presentation components 402 define the appearance and behavior of the component applications 105 as it [is] displayed by the user interface 202”), [0045] (“workflow components 406 of the component application 105 define processing that occurs when an action is to be performed”).)

181. For example, *Shenfield* discloses that its component applications can “be executed on native code” to display content “in the terminal runtime environment 206 provided by [a number of] device[s] 100.” (Ex-1007, ¶¶[0026], [0029]). *Shenfield* explains that one way to achieve this interoperability is to tailor components to the platform of particular devices, stating that its “presentation components 402 [of component applications 105] may vary depending on the client platform and environment of the device 100” and that it provides “bundle[s] of platform-neutral ... component descriptors” and “platform-specific presentation component 402 descriptors for various predefined client runtimes (*i.e.*, specific runtime environments 206—see FIG.2).” (*Id.*, ¶[0044]; *see also id.* (“different sets

of presentation components 403a, 403b, 403c, representing the different supported user interfaces 202 of the devices 100.”.) *Shenfield* further discloses that “workflow components 406 are written as a series of instructions” that operate “as a set of rules for operations on the other components” and discloses “defin[ing] differing work flows based upon different supported capabilities or feature of particular devices.” (*Id.*, ¶[0045].) *Shenfield* shows that each of its data 400, presentation 402, message 404, and workflow 406 components are separate sets of code in Figure 4a:



Based on the above disclosures, a POSITA would have understood that *Shenfield*’s platform-specific presentation and workflow components are computer code that is written to operate specifically in a particular device platform and, when executed,

provides instructions to the device. A POSITA would have further understood that these components are separate from the platform-neutral components.

(ii) A POSITA Would Have Been Motivated to Modify *Huang*'s System with *Shenfield*'s Device-Specific Components with a Reasonable Expectation of Success

182. It is further my opinion that a POSITA would have looked to *Shenfield* to improve *Huang*'s system, would have been motivated to modify the system of *Huang* in view of the disclosures of *Shenfield*, and would have had a reasonable expectation of success to make such modifications.

183. A POSITA would have looked to *Shenfield* when considering the implementation of *Huang*'s system as of the invention date. For example, a POSITA would have been aware that many less-powerful devices that were capable of accessing the internet, such as cellular telephones and PDAs, had become popular technologies that presented issues regarding computing resources that made other more resource-intensive solutions insufficient for such devices. (Ex-1007, ¶¶[0002]-[0003].) When paired with *Huang*'s suggestion to explore alternative configurations of *Huang*'s system (*e.g.*, Ex-1005, ¶[0008]), a POSITA would have looked to other teachings that maintain interoperability but allow for more efficient processing on such smaller devices. A POSITA therefore would have looked to *Shenfield* in particular because it offers a solution to this specific problem of addressing the

resource constraints of myriad, less-powerful, internet-accessible devices. (*E.g.*, Ex-1007, ¶¶[0002]-[0003], [0054]-[0056].)

184. A POSITA would have sought to identify options that preserve the ability of *Huang*'s browser-based applications to operate on a wide array of devices without relying on web browsers. A POSITA would have recognized that *Shenfield* discloses such a configuration, providing "the efficiency of native application execution" (Ex-1007, ¶[0054]), "an effective data storage and persistence model" (Ex-1007, ¶[0055]), and "a platform neutral model" as in *Huang* that allows for "the principle 'write once run everywhere'" (Ex-1007, ¶[0056]). Thus, a POSITA would have looked to *Shenfield* in particular because it offers a solution to the exact issues presented by the proliferation of less-powerful, internet-accessible devices, including greater efficiency and data persistence to address these resource constraints, while maintaining the advantage of *Huang*'s system that allows its applications to run across various devices.

185. A POSITA would have also been motivated to modify *Huang*'s designer 104 with *Shenfield*'s solution of "platform-specific" presentation and workflow components. To implement such components would optimize *Huang*'s applications to avoid the efficiency and data persistence issues that strain the computing resources in smaller Internet-accessible devices. (Ex-1007, ¶¶[0002]-[0003].) Further, *Shenfield*'s disclosures that native applications "provid[e] a

relatively optimized application program for each runtime environment” over “resource intensive” solutions would have motivated a POSITA to implement *Shenfield’s* solution. (Ex-1007, ¶[0005].) *Shenfield* further discloses that its “component applications 105 may be executed as native code or interpreted by another software module,” which a POSITA would have understood includes the ability to operate within web browsers, without sacrificing the “write once run everywhere” characteristic of *Huang’s* applications. (Ex-1007, ¶¶[0029], [0056].) Accordingly, a POSITA would have been motivated to implement *Shenfield’s* device-specific components, which are the *Player that is a device-dependent code*, in *Huang’s* system to achieve these benefits that would allow for optimizing the performance of *Huang’s* applications on a wider range of devices. Specifically, a POSITA would have been motivated to substitute *Huang’s* players with *Shenfield’s* players written in device-specific code that can provide instructions tailored to optimize the display of content specified in *Huang’s* device-independent applications for each runtime environment, thereby optimizing performance and improving user experience for a wider audience of end users. (Ex-1007, ¶¶[0005], [0029], [0056]; *see also id.*, ¶[0054].)

186. In my opinion, a POSITA would have had a reasonable expectation of success in modifying the system of *Huang* with *Shenfield’s* device-dependent components. Exemplary reasons are set forth below.

187. For example, *Shenfield*'s device-specific components are designed to operate with *Shenfield*'s device-independent components, which are written in a similar code to *Huang*'s applications (*Huang*'s XML definitions (Ex-1005, ¶¶[0008], [0042], [0059]) versus *Shenfield*'s "structured definition language" (Ex-1007, ¶[0040]), including XML (*id.*, ¶[0046])). Put differently, *Shenfield*'s device-specific components (*player*) would have been compatible with *Huang*'s device-independent applications because *Shenfield*'s device-independent components utilize similar code as *Huang*'s applications such that modifying *Huang*'s system to use *Shenfield*'s *player* would have been relatively straightforward to apply.

188. Further, *Shenfield*'s platform-specific presentation and workflow components operate similarly to the functionality of *Huang*'s *player*. *Shenfield* explains that its components that can be designed to be platform dependent include (1) presentation components that "define the appearance and behavior of the component application 105 as it [is] displayed by the user interface 202," such as "specify[ing] GUT screens and controls, and actions to be executed when the user interacts with the component application 105 using the user interface 202" (Ex-1007, ¶[0043]); and (2) workflow components that "define processing that occurs when an action is to be performed, such as an action specified by a presentation component 402," are "a series of instructions in the selected programming/scripting language ... [that] can be compiled into native code and executed by the runtime

environment,” and “supports a correlation between the messages and defines application flow as a set of rules for operations on the other components” (*id.*, ¶[0045]). Put differently, the platform specific components relate to the presentation of a user interface and the instructions for executing the actions necessary to operate within a runtime environment. Thus, *Shenfield*’s platform specific components correspond to *Huang*’s player that executes in a browser, as explained in *Nachnani*, which states that “[a]n application can be divided into at least two main parts: a user interface, which is presented to a user and interacts with the user, and business logic, which performs the tasks required of the application, in accordance with user input received from the user interface. The web browser runs the user interface.” (Ex-1006, ¶[0005].) Thus, *Shenfield*’s platform-specific components would have been able to execute *Huang*’s application in the runtime environment of a device and provide instructions relating to the display and other necessary functionalities.

189. Further, a POSITA would have been able to modify *Huang*’s designer, which already is a program that generates code, to generate these platform-specific components. This would have been an obvious choice because it would offer the benefit of efficiently leveraging an already existing resource in *Huang*’s system that performs essentially the same function of generating code. (*See, e.g.*, Ex-1005, ¶[0042].) Indeed, a POSITA would have understood that *Huang* suggests that its designer already generates a player because *Huang*’s system requires a player and

the designer already generated code in the form of applications, thus indicating that it can also produce code in the form of players. Further, *Shenfield* expressly discloses that its system generates all components. (See Ex-1007, ¶[0057].) Altering *Huang*'s system to generate device-specific players, as disclosed in *Shenfield*, that allow for the presentation of *Huang*'s application would have been a simple substitution of known elements to achieve predictable results.

190. Thus, it is my opinion that a POSITA would have found *producing a Player*, where said *Player* is a device-dependent code obvious over *Huang* in view of *Shenfield*.

- h. [1.g] “such that, when the Application and Player are provided to the device and executed on the device, and when a user of the device provides one or more input values associated with an input symbolic name to an input of defined UI object, 1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service, 2) the web service utilizes the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name, 3) said Player receives the output symbolic name and corresponding one or more output values and provides instructions for a display of the device to present an output value in the defined UI object.”**

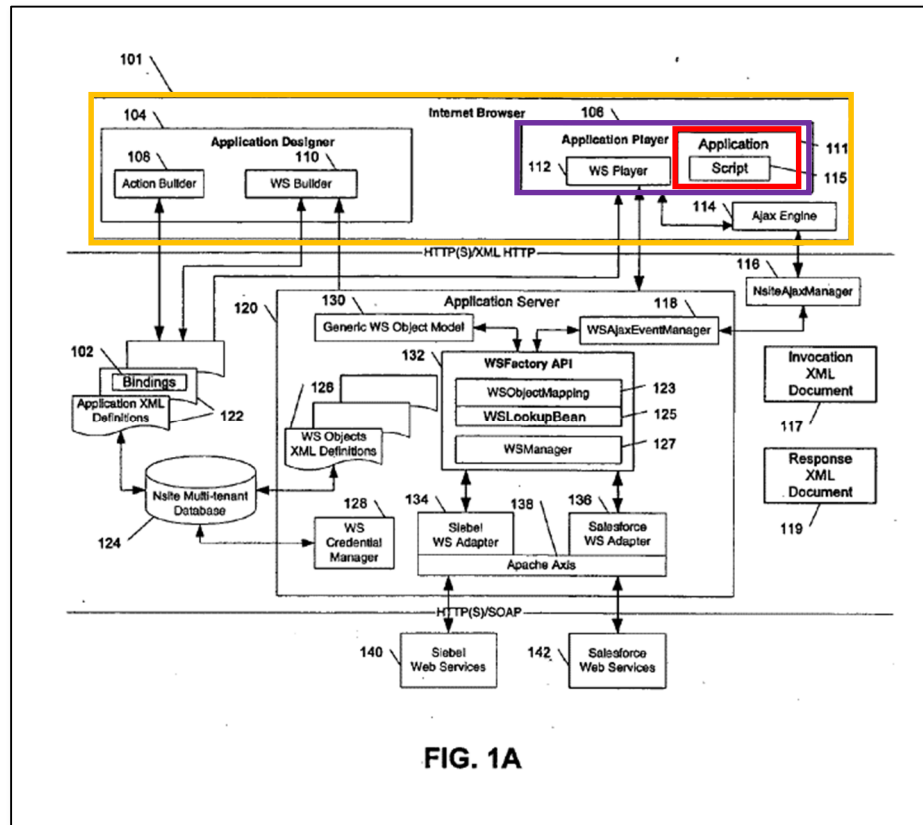
191. It is my opinion that a POSITA would have found this element obvious over *Huang* in view of *Shenfield*. The district courts construed the terms “Application” (“device-independent code containing instructions for a device and which is separate from the Player”), “Player” (“device-specific code which contains

instructions for a device and which is separate from the Application”), “device-dependent code” (“code that specific to the operating system, programming language, or platform of a device”), and “device-independent code” (“code that is not specific to the operating system, programming language, or platform of a device”). (*Supra* Section VII.) As discussed below, *Huang* and *Shenfield* together disclose or suggest each aspect of element [1.g] and thus renders it obvious.

- (i) “such that, when the Application and Player are provided to the device and executed on the device, and when a user of the device provides one or more input values associated with an input symbolic name to an input of defined UI object,

192. In my opinion, *Huang*, as modified by *Shenfield*, discloses that the application and player are provided to a device and executed on that device. *Huang* states that its player is provided to the Internet browser on a device and the application is loaded into the player and executed. (*E.g.*, Ex-1005, ¶¶[0040] (“The Player 106 is a computer program that provides a user interface which executes a browser based application.”), [0041] (“Player 106 [is] provided to the Internet Browser 101 by a server 120 as web pages.”), [0042] (“Applications created by the Designer 104 are ... loaded into the Player 106 as the application 111.”); *see also* Ex-1006, ¶[0014] (“providing a software application” and “providing a web browser and providing a user interface to a user on the web browser from the received representation and data”).) *Huang* also explains that “each element or component

shown in FIG. 1A is a software computer program entity to be executed on a computer” (Ex-1005, ¶[0040]), and Figure 1A of *Huang* shows that both the **application** and **player** are provided to the same **Internet browser**:



Huang explains that “**Player 106** [is] *provided to* the **Internet Browser 101** by a server 120 as web pages” and that “**Applications** created by the Designer 104 are ... *loaded into the Player 106* as the application 111.” (Ex-1005, ¶[0041]-[0042]; see also Ex-1006, ¶[0014], [0097].) *Huang* also explains that “[e]ach element or component shown in FIG. 1A is a software computer program entity to be executed on a computer”; that “Player 106 execute[s] in a conventional Internet Browser 101”; and that “Player 106 is a computer program that provides a user interface which

executes a browser based application.” (Ex-1005, ¶[0040].) A POSITA would have understood that *Huang*’s application and player are provided to the device for execution because the player is provided to and executes in a browser that runs on a device and the application is “loaded into the Player 106” in order to run.

193. *Shenfield* similarly discloses providing both platform-neutral data, which are analogous to *Huang*’s applications, and platform-specific components, which are analogous to *Huang*’s player. The platform-neutral components include the data components, which define the content much like *Huang*’s applications, and the platform-specific components include the presentation and workflow components, which present the content on a display like *Huang*’s player. (Ex-1007, ¶¶[0039]-[0045].)

194. From the above disclosures, a POSITA would have understood that both the application and player of the modified *Huang-Shenfield* system are provided to the device for execution.

195. It is also my opinion that *Huang* discloses that when the device executes the player and application, the device presents a user interface containing objects that allows *a user of the device to input one or more values of a defined UI object*. Specifically, *Huang* discloses that “Player 106 executes browser-based applications 111, which have application user interface components, such as text fields, for receiving data values from a user.” (Ex-1005, ¶[0054].) *Huang* identifies an

example of a “text input component”—the “OppNameTextArea” application component. (Ex-1005, ¶[0103].) *Huang* further states that applications are able to interact with web services and exchange data values through various interactions. (*Id.*, ¶[0054] (“[A]pplications executing in the Player 106 [are able to] interact with web services by, for example, looking up a value from a web service and storing the value in an application component associated with the application 111, or inserting, updating, or deleting a value in a web service object, where the value is specified by an application component.”); *see also id.*, ¶[0058] (“[A] set of bindings 102 is associated with a user interface component, such as button or link, that, when selected by a user, causes a web service interaction such as a lookup or insert. ... Player 106 ... can invoke any type of web service interaction, including lookups, inserts, updates, and deletes. Data values will be transferred between the application components and web service attributes specified in the bindings 102.”).) *Huang* further explains that “[w]hen a user submits data values in an application, the WSManger 127 inserts into or updates the web service using the new values” or if a “user requests data in an application, the WS Manager 127 performs a lookup operation to query the web service for the data.” (Ex-1005, ¶[0049].) *Huang* further explains that the value of a presentation component “is displayed to a user and may be set by a user, e.g., as a value of a text box or of a drop-down list.” (Ex-1005, ¶[0057].) A POSITA would have understood from these disclosures that when the

player and application execute on a device, the user is presented a display that includes defined user interface objects that allow a user to input values.

196. *Nachnani* further confirms that the executed player and application allows a user to provide an input value to an input of a defined UI object. Specifically, *Nachnani* explains that, when “[a] web browser 132 executes a Player UI 134, which in turn[] executes an application 135,” “[t]he Player UI 134 presents the application 135 to a user 130 via the web browser 132.” (Ex-1006, ¶[0040].) *Nachnani* continues that “[t]he application 135 includes at least one component 136, which represents a user interface element such as a text box or a web link” and that “[t]he component 136 also includes a data value 142, e.g. a text string or numeric value, ***which can be specified by a user via the Player UI 134.***” (*Id.*) *Nachnani* also states that “Application Data 120 may be provided by the user 118 when the Player UI 114 executes the application.” (*Id.*, ¶[0039].) These disclosures further confirm that a user may provide an input value to an input of a defined UI object when the player and application are executed.

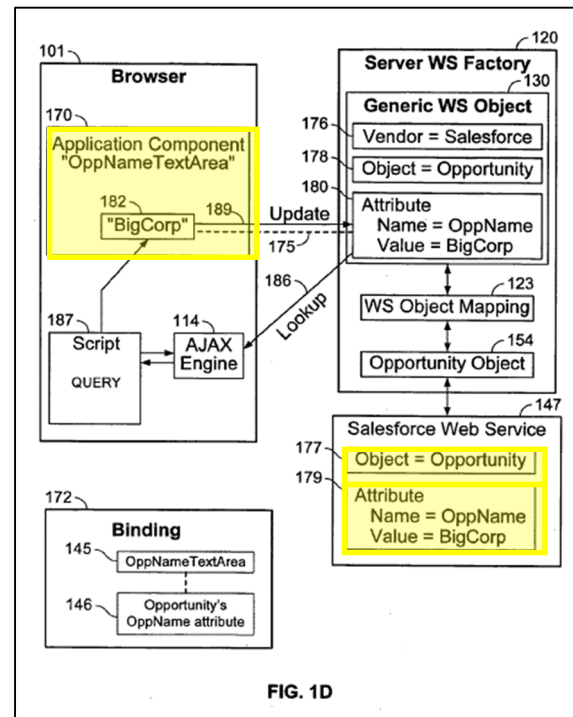
197. Likewise, *Shenfield* also discloses that its presentation components allow for user input, such as enabling a user to “type[] in an edit box or push[] a button.” (Ex-1007, ¶[0043].) Accordingly, *Shenfield* discloses that a user may provide one or more input values to an input of a defined UI object.

- (ii) 1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service.”

198. It is also my opinion that *Huang* discloses that, when a user provides one or more input values, the input value is associated with a symbolic name and that *the device provides user provided one or more input values and corresponding input symbolic name to a web service*. As discussed above, *Huang* discloses that its application components, which may be a text area such as the “OppNameTextArea” in Figure 4B, are bound to a web service attributes identified by symbolic names (such as those listed under “Data Source in Figure 4C). (*Supra* Sections X.B.1.c(ii), X.B.1.e.) As previously discussed, *Huang* explains the bindings between the application components and web service attributes allow for the exchange of data to and from a web service, stating that web service attributes are “values that can be stored in or retrieved from web services” and that bound application components can “set and get values of the vendor-specific attribute 155.” (Ex-1005, ¶¶[0055], [0057]-[0058], [0106]-[0107], Figs. 4B-4C; *see also id.*, ¶[0107] (“establish bindings between application components (also referred to herein as controls), and attributes of the selected web service object 416”).) A POSITA would have understood from *Huang*’s disclosures that *Huang*’s system associates the input value to the bound symbolic name, and both are sent to the web service because a POSITA would understand that raw data without any identifying information could not be

used by a web service because the web service would have been unable to identify to what attribute the value relates.

199. Figure 1D of *Huang* confirms this understanding, depicting that the input value and the associated symbolic name are sent to the web service:



(Ex-1005, Fig. 1D.)

As can be seen in the above figure, the symbolic name of the web service object (“Opportunity”), the symbolic name of the web service attribute (“OppName”), and the associated input value (“BigCorp”) is sent to the web service.

- (iii) “2) the web service utilizes the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name,”

200. In my opinion, *Huang* discloses or suggests that *the web service utilizes the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name.* As discussed in the immediately preceding section above, *Huang* discloses a user inputting a value for an attribute of a web service object, and this input value is associated with a symbolic name and both the input value and symbolic name are sent to the web service. As will be discussed below, *Huang* discloses that the web service utilizes both the input value and the input symbolic name to generate one or more output values having an associated output symbolic name.

201. In my opinion, *Huang* discloses that the web service utilizes the input value and associated input symbolic name to generate an output value having an associated output symbolic name. *Huang* discloses that an output of a web service may be dependent on an input value—i.e., that the input is a factor that determines the relevant output. (*Id.*, ¶[0062] (“A user of the WS Player 112 can then select a value from the drop-down menu to cause the WS Player 112 to retrieve objects for which the corresponding attribute is equal to the selected value from the web service.”).) A POSITA would have understood that, in this example, an input value is provided to the web service and this value is associated with an input symbolic name, and the web service utilizes the input symbolic value and the associated input symbolic name because the output from the web service that is returned to the player

are those “objects for which the corresponding attribute is equal to the selected value from the web service.” (*Id.*) *Huang* further discloses that its system, when processing a request, “extract[s] the input data for the lookup,” which includes the “web service object name.” (Ex-1005, ¶[0129].) A POSITA would have understood that *Huang*’s system thus passes along the input data and the web service object name, which would be the symbolic name, to the web service, which uses this information to generate an output.

202. Further, a POSITA would have understood that a web service would necessarily require and utilize data inputs that both include a value and an identifier, which in the case of *Huang* is a symbolic name. A web service would be unable to have understood how to process or account for input data if the incoming message did not specify, for example, the specific web component that was being invoked or an identifier indicating that a data value relates to a particular required input. Put differently, if *Huang*’s system provided data without its input symbolic name, the web service would be unable to understand what the data means or how to utilize the data in executing its functions.

- (iv) “3) said Player receives the output symbolic name and corresponding one or more output values and provides instructions for a display of the device to present an output value in the defined UI object.”

203. It is also my opinion that *Huang* discloses that its *player receives the output symbolic name and associated output values* from a web service and, in turn,

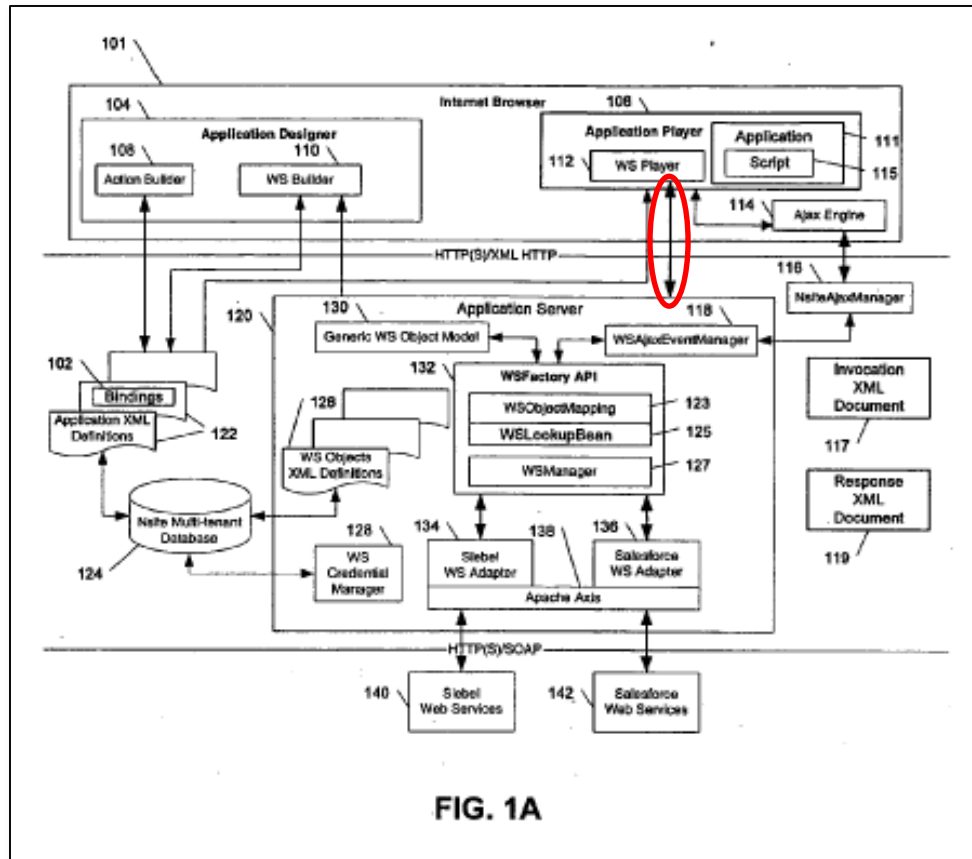
provides instructions for a display of the device to present that output value in the defined UI object. As discussed above, a POSITA would have understood that *Huang*'s system displays content to a user utilizing web components that can receive user inputs or output data. (*Supra* Section X.B.1.a.) For example, *Huang* discloses that the values of its application components, which include the outputs from web services, are displayed to the user. (Ex-1005, ¶¶[0057] (“The application component includes a value 192, which is displayed to a user”), [0070] (“query the web service and **display a list of one or more values returned** by the web service according to the binding”).) As discussed below, *Huang* discloses that its player receives the output from the web service and provides the instructions to display this output.

204. For example, *Huang* discloses that its player is what enables “applications executing in the Player 106 to interact with web services by, for example, looking up a value from a web service and storing the value in an application component associated with the application 111, or inserting, updating, or deleting a value in a web service object, where the value is specified by an application component.” (Ex-1105, ¶[0054].) *Huang* contains numerous disclosures that state that its player is responsible for executing the communication with web services. (*Id.*, ¶¶[0049] (“The WSManger 127 may be invoked ... by the Player 106, e.g., to submit or lookup data values.”), [0057] (“Player 106 use[s] the Generic WS Object 130 to interact with the vendor web service 156, e.g., to set and

get values of the vendor-specific attribute 155.”), [0062] (“A user of the WS Player 112 can then select a value from the drop-down menu to cause the WS Player 112 to retrieve objects for which the corresponding attribute is equal to the selected value from the web service.”); *see also id.*, ¶[0040] (“Player 106 is a computer program that provides a user interface which executes a browser based application 111”), [0063] (“The bindings 102 are used by the WS Player 112 when a web service interaction is performed”), [0072] (“the WS Player 112 uses binding information that establishes a mapping between WS object attributes and application components to display a Web Service lookup user interface for storing or retrieving the values of those components to or from web services.”).)

205. *Huang* also states that its systems “sends the XML data as a response, e.g., to a client such as the Internet Browser 101 of FIG. 1A, or to software running in the browser, such as ... the Player 106 of FIG. 1A, which delivers the data to the application.” (Ex-1005, ¶[0129].) Because the modified *Huang-Shenfield* system would utilize a player that operates in the native runtime environment and would not rely on a web browser, a POSITA would have understood that the runtime player in the modified *Huang-Shenfield* system would receive the output of the web service.

206. *Huang* further confirms that its player interacts with web services in Figure 1A, which depicts the WS Player component of *Huang*’s player interfacing with web services:



(Ex-1005, Fig. 1A.)

207. From these disclosures, a POSITA would have understood that *Huang*'s player mediates the communications between the application component and the web service and, therefore, *Huang*'s player receives the outputs of a web service. From these disclosures, a POSITA would have understood that *Huang*'s player also displays the output from these web services. (*E.g.*, Ex-1005, ¶[0072] ("the WS Player 112 uses binding information that establishes a mapping between WS object attributes and application components to display a Web Service lookup

user interface for storing or retrieving the values of those components to or from web services”).)

208. *Nachnani*, which as discussed above *Huang* incorporates by reference, confirms that *Huang*’s player “presents the application 135 to a user 130 via the web browser.” (Ex-1006, ¶[0040].)

209. Accordingly, a POSITA would have understood that *Huang*’s player displays the output of a web service. A POSITA would further have understood that *Huang*’s player accomplishes this by providing instructions to the web browser because a computer program is a set of instructions and providing instructions is the only way to cause a device to perform any functionality.

210. *Huang* further discloses “text fields” (Ex-1005, ¶[0054]), “text box[es]” (*id.*, ¶[0058]) and “text area component[s]” (*id.*, ¶[0067]). In my opinion, a POSITA would have understood that these components were capable of both receiving input text and displaying output text. At a minimum, a POSITA would have found it predictable to use the same application component for input and output to leverage existing resources in an efficient manner. Using these components for both input and output would have been a predictable design choice.

211. To the extent *Huang* does not explicitly disclose that *Huang*’s player receives the output symbolic name and output value from a web service and provide instructions to present the output value on a display of a device, a POSITA would

have found this obvious over *Huang* in view of *Shenfield*. *Shenfield* explains that its “workflow components” are “a series of instructions in the selected programming/scripting language” and can be “based upon different supported capabilities or feature of particular devices.” (Ex-1007, ¶[0045].) *Shenfield* further explains that its “workflow components” define “[p]resentation workflow and message processing.” (*Id.*) A POSITA would have understood that the workflow components of *Shenfield*, in operation, would provide instructions for carrying out messaging between its component application and a web service and provide instructions to display data to a user, such as data obtained from a web service. Put differently, the workflow component would have been understood as processing messaging—i.e., both sending and receiving messages from a web service—and processing the data for presentation on a display. I refer to my prior discussion regarding the motivation to combine *Huang* and *Shenfield* with a reasonable expectation of success, and I incorporate it by reference as if set forth herein. (*Supra* Section X.B.1.b.)

2. Claim 3

- a. “The system of claim 1, where said web component is a text chat, a video chat, an image, a slideshow, a video, or an RSS feed.”**

212. In my opinion, *Huang* in view of *Shenfield* renders this claim obvious. As discussed above, it is my opinion that *Huang* in view of *Shenfield* renders claim

1 obvious. (*Supra* Section X.B.1.) It is further my opinion that the modified *Huang-Shenfield* system renders obvious a *web component that is a text chat, a video chat, an image, a slideshow, a video, or an RSS feed*. I understand that these elements are stated in the alternative and, therefore, satisfying one of a text chat, a video chat, an image, a slideshow, a video, or an RSS feed would render this claim obvious.

213. In my opinion, *Shenfield* discloses that a *web component is an RSS feed*. *Shenfield* discloses that “[t]he implementation of mappings 804 facilitates building the wireless applications 302 based on server-to-device notifications.” (Ex-1007, ¶[0071].) *Shenfield* continues that “[t]hese mappings 800 can be applicable for a variety of wireless applications 105 such as stock trading, news updates, alerts, [and/or] weather updates.” (*Id.*) A POSITA would have understood that such updates can be implemented as RSS feeds, which is a well-known format for sending XML-formatted plain text as updates.

214. It is further my opinion that *Shenfield* discloses or suggests web components that are text chats and images. *Shenfield* discloses that its workflow component can render “objects that represent ... pop-ups, dialog boxes, [and] text areas”; that it can also provide for “image loading”; and that its system “combines user interface elements and fixed and computed text and images, and is reactive to user interaction on the user interface 202.” (Ex-1007, ¶[0078].) A POSITA would have understood that pop-ups, dialog boxes, and text areas” are elements useful for

and appropriate to a text-chat interface. (*See, e.g.*, Ex-1023, ¶[0005] (“[E]ach user being able to input text material intended to be conversational in nature. The conversational input from, e.g., a first user is relayed to the computers of the other users who also happen to be logged onto the chat room, such that the text from the first user is presented to the other users. Then, the other users can respond if they like by inputting text material of their own, and their text material is likewise related to the other ‘occupants’ of the chat room, including the first user.”).) A POSITA would have also understood that “image loading” would mean that *Shenfield*’s system obtains and renders images from third-party sources, such as those provided through web services. (*E.g.*, Ex-1024, ¶[0006] (“a server hosting a media storage and display service”).) A POSITA would further have understood the disclosures of “combin[ing] user interface elements” with “images” discloses an image obtained from a web component because user interface elements are the elements identified as bound to web components, thereby indicating that the user interface elements here are used to obtain images from a web component and render them on the display.

215. At a minimum, a POSITA would have found binding the application components in the system of *Huang*, as modified by *Shenfield*, to a text chat, image or RSS feed web service obvious. Web components for text chat, images, and RSS feeds were well known in the art and were among the finite set of web-service options that were available to a POSITA. Well-known web services as of April 2008

include text communication or chat services (*e.g.*, Ex-1021, ¶[0002] (“Chat services on the Internet provide for real time communication between two users via a computer, wireless device, or any other text based communication apparatus.... Most networks and online services offer some type of chat feature.”); Ex-1022, ¶[0004] (“Web service interface (WSI) messaging”); Ex-1023, ¶[0005] (“Another popular feature of the Internet are so-called ‘chat rooms’. Essentially, a chat room is a computer site that can be accessed (*i.e.*, ‘logged onto’) simultaneously by many users, with each user being able to input text material intended to be conversational in nature.”)), image services (*e.g.*, Ex-1024, ¶[0006] (“a server hosting a media storage and display service”)), and RSS feeds (*e.g.*, Ex-1025, ¶[0003] (“Generally, RSS provides web content or summaries of web content together with links to the full versions of the content, and other meta data.”)). Because web services are designed to be accessed, binding an application to such web services would have been a simple implementation of known technologies via standard means to achieve predictable results. Indeed, *Huang* and *Nachnani* contain disclosures that would have been understood as identifying components and functions that are readily suited for text chat, image, or RSS feed web services, such as text input fields, text output fields, and image application components that can be bound to web service inputs and outputs. (*E.g.*, Ex-1005, ¶¶[0057], [0068]; Ex-1006, ¶[0042], Fig. 2a.)

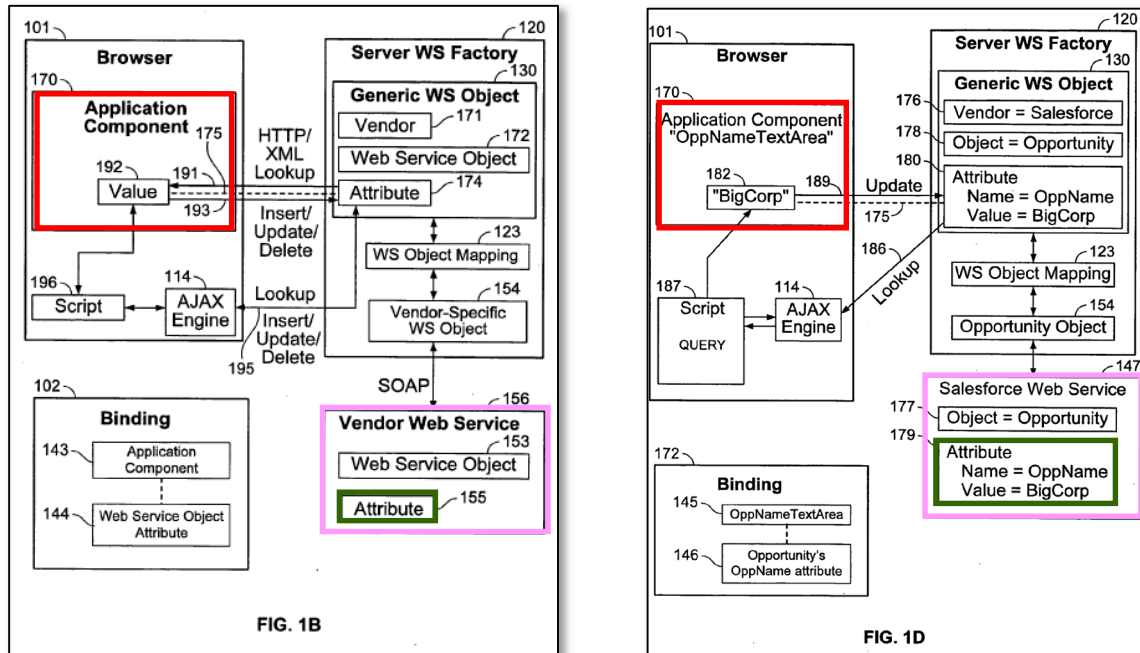
216. Accordingly, it is my opinion that *Huang* in view of *Shenfield* discloses, suggests, and renders obvious claim 3.

3. Claim 5

a. “The system of claim 1, where said UI object is an input field for a web service.”

217. In my opinion, *Huang* in view of *Shenfield* renders claim 5 obvious. As discussed above, it is my opinion that *Huang* in view of *Shenfield* renders claim 1 obvious. (*Supra* Section X.B.1.) *Huang* discloses the requirement that the *UI object is an input field for a web service* for the reasons discussed above.

218. Specifically, *Huang* discloses a text input field for a web service in the form of the “OppNameTextArea” application component. (Ex-1005, ¶¶[0054], [0103].) *Huang* explains that **application components**, such as “OppNameTextArea” may be used to “insert or update” **web service attributes**. (*Id.*, ¶¶[0056], [0064], [0067].)



(Ex-1005, Figs. 1B, 1D.)

219. Thus, *Huang* explains that its application components can be input fields for a web service, providing at least one specific example (the “OppNameTextArea” application component) that is a text input field for a web service.

4. Claim 6

- a. “The system of claim 1, where said UI object is an input field usable to obtain said web component, where said input field includes a text field, a scrolling text box, a check box, a drop down-menu, a list menu, or a submit button.”

220. In my opinion, *Huang* in view of *Shenfield* renders claim 6 obvious. As discussed above, it is my opinion that *Huang* in view of *Shenfield* renders claim 1

obvious. (*Supra* Section X.B.1.) *Huang* discloses the requirement that the *UI object* is an input field usable to obtain said web component, where said input field includes a text field, a scrolling text box, a check box, a drop down-menu, a list menu, or a submit button, for the reasons discussed above. I understand that a text field, a scrolling text box, a check box, a drop down-menu, a list menu, or a submit button are stated in the alternative and so long as one element is rendered obvious, the claim is rendered obvious.

221. *Huang* discloses text fields and submit buttons, as discussed in detail above. X.B.1.c (discussing the “OppNameTextArea” text input field and “button” components for invoking web services).) *Huang* further discloses that a UI object can be a “**drop-down menu** that includes a list of possible values for [an] attribute,” which, in this example, is an input to a web service used to “**retrieve** [only] objects for which the corresponding attribute is equal to the selected value,” thereby disclosing the additional requirement of claim 6. (Ex-1005, ¶[0062].) Thus, *Huang* expressly discloses at least a drop-down menu, text input fields, and submit buttons.

222. I further note that *Nachnani* further confirms that *Huang* discloses such input fields. For example, *Nachnani* states that the component may be, among other things, a “Text Box, Text Area, Radio Button, Check Box, Select List (e.g. a drop-down list), ... [or] Button,” describing such components in the table below (Ex-1006, ¶[0042]):

| COMPONENT TYPE | DESCRIPTION | PROPERTIES |
|----------------|--|---|
| Text Box | A text input field for reading data of a specific type, e.g. string or numeric data. Contains a text value. | Name, Required, Readonly, Keyword Field, Data Type, Allowed Characters, Visibility Privileges, Edit privileges, Routing edits (enable or disable), data type, data format, max characters, default value, action (event handler). |
| Text Area | A text edit box for editing a text string. Can display multiple rows of text and allow the user to edit the text. Contains a text value. | Name, required, read only, keyword field, visibility and edit privileges, routing edits (enable or disable), allowed characters, data format, max characters, max rows, max chars, default value, action (event handler). |
| Date Picker | A data selector. Contains a date value. | Name, required, read only, keyword field, visibility and edit privilege, routing edits (enabled or disabled), data format, default value, action (event handler). |
| Check Box | A check box with two possible values (checked or unchecked). Contains a boolean value. | Label, name, edit and visibility privileges, routing edits (enable or disable), initial state (checked or unchecked), action (event handler). |
| Radio Buttons | A set of buttons, one of which can be selected. Contains a value representing a selected button. | Name, radio group, edit privileges, routing edits, initial state, checked value, action (event handler). |
| Combo Box | A drop-down box for selecting one item from a list of items. | Name, required, keyword field, visibility and edit privileges, routing edits (enable or disable), data type, height, allow multiple selections (yes or no), data lookup source, action (event handler). |
| Label | A text label for displaying information. Does not contain a value. | Name, layout style, alignment. |
| Button | A button that can be selected by a user to cause an event handler to be called. | Label, name, visibility and edit privileges, action (event handler), web service, link. |
| Attachment | A file associated with the application. | Name, attachment type, visibility privileges, action privileges. |
| Link | A web link that refers to a web page, web service, or other resource. When the link is selected in the Player UI, the web page, web service, or other resource will be accessed. | Label, name, visibility and edit privileges, action (event handler), web service, link (URL). |

5. Claim 7

- a. **“The system of claim 1, where said web component is an output of a web service, is the text provided by one or more simultaneous chat sessions, is the video of a video chat session, is a video, an image, a slideshow, an RSS display, or an advertisement.”**

223. In my opinion, *Huang* in view of *Shenfield* renders claim 7 obvious. As discussed above, it is my opinion that *Huang* in view of *Shenfield* renders claim 1 obvious. (*Supra* Section X.B.1.) Also discussed above for claim 3, it is also my opinion that *Huang* in view of *Shenfield* renders obvious a *web component that is a*

text chat, a video chat, an image, a slideshow, a video, or an RSS feed. (*Supra* Section X.B.2.) Accordingly, it is my opinion that *Huang* in view of *Shenfield* renders obvious that *said web component is an output of a web service, is the text provided by one or more simultaneous chat sessions, is the video of a video chat session, is a video, an image, a slideshow, an RSS display, or an advertisement* for the same reasons that *Huang* in view of *Shenfield* renders claim 3 obvious.

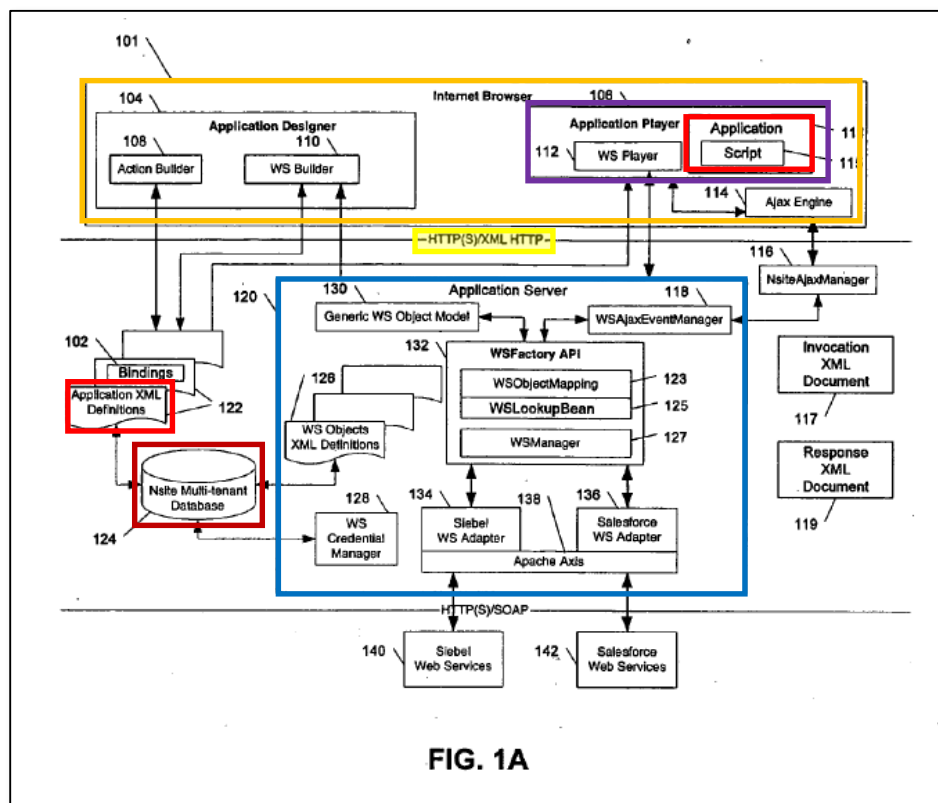
6. Claim 11

a. “The system of claim 1, where said code is provided over said network.”

224. In my opinion, *Huang* in view of *Shenfield* renders claim 11 obvious. As discussed above, it is my opinion that *Huang* in view of *Shenfield* renders claim 1 obvious. (*Supra* Section X.B.1.) I am informed that “said code” may refer to the code that is generated by the system (i.e., “A system for generating code”), the “device-independent code” of the “Application,” and/or the “device-dependent code” of the “Player” recited in claim 1. (*See* Ex-1001, Claim 1.) In my opinion, the code generated by the modified *Huang-Shenfield* system, including both the application and player, are provide over a network because both *Huang* and *Shenfield* disclose providing its application and player over a network.

225. *Huang* discloses that “**Player 106** [is] provided to the **Internet Browser 101** by a **server** as web pages” (Ex-1005, ¶[0041]) and that **applications** “are stored

... in a database 124, and loaded into the Player 106 as the application 111” (id., ¶[0042]).

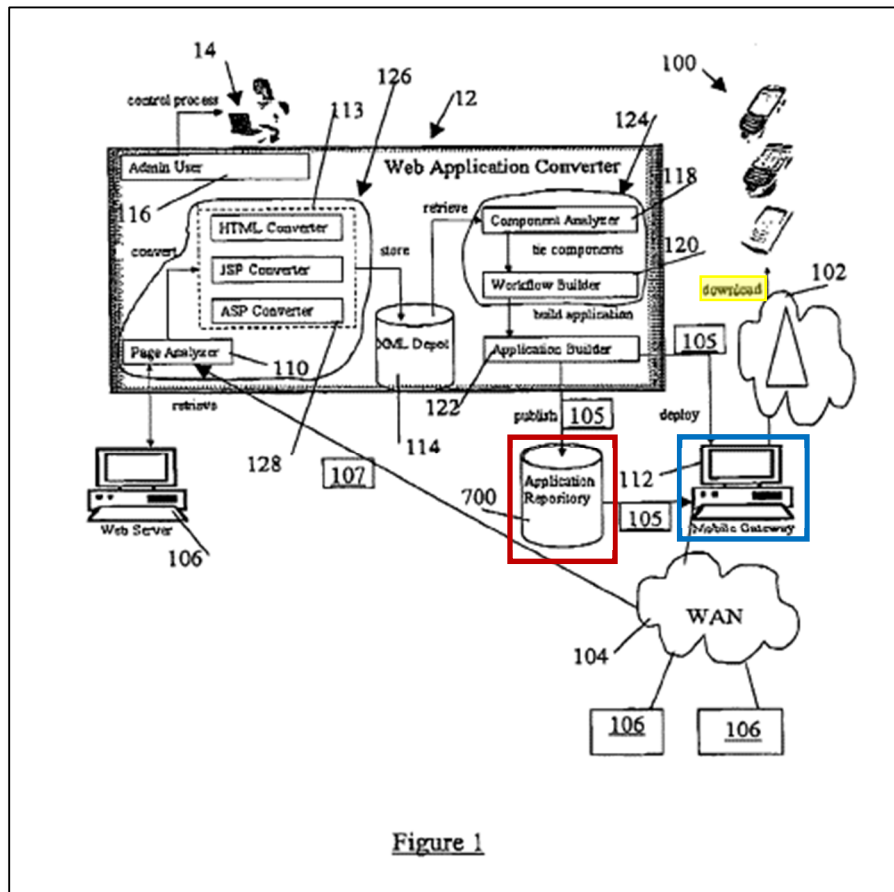


(Ex-1005, Fig. 1A.)

226. A POSITA would have understood that *Huang* provides its code in the form of applications and players over a network in view of Figure 1A identifying an “HTTP(S)/XML HTTP” protocol between the browser 120 and database 124/server 120.

227. *Shenfield* also discloses providing its code in the form of its component applications (which include both its platform-neutral and platform specific components) over a network. *Shenfield* states that “application definitions can be

hosted in the **repository 700** as a bundle of platform-neutral component definitions linked with different sets of presentation components 403a, 403b, 403c, representing the different supported user interfaces 202 of the devices 100” and that a user “makes a ... download request message to the **server 112**” to download these components via a network. (Ex-1007, ¶[0044].)



(Ex-1007, Fig. 1.)

228. A POSITA would have understood that *Shenfield* provides its code to devices over “wireless network 102” and, therefore, expressly discloses that it provides its code over a network. (Ex-1007, ¶[0021].)

229. In view of both *Huang* and *Shenfield* disclosing that they provide their code over networks to the device of the user, it is my opinion that the modified *Huang-Shenfield* system would do the same.

7. Claim 12

- a. [12.pre] “A method of displaying content on a display of a device utilizing a registry of one or more web components related to inputs and outputs of a web service obtainable over a network, where each web component includes a plurality of symbolic names of inputs and outputs associated with each web service, and where the registry includes: a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and b) the address of the web service, said method comprising:”**

230. I understand that the phrase “A method of displaying content on a display of a device utilizing a registry of one or more web components related to inputs and outputs of a web service obtainable over a network, where each web component includes a plurality of symbolic names of inputs and outputs associated with each web service, and where the registry includes: a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network, where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service, and b) the address of the web service, said method

comprising:” is the preamble to claim 12 of the ’755 patent. I have been asked to assume that the preamble is a claim limitation. Under that assumption, in my opinion *Huang* in view of *Shenfield* renders these features obvious for many of the same reasons that claim 1 is obvious. (*Supra* Section X.B.1.)

231. For clarity, I separately address each aspect of the preamble of claim 12.

- (i) “A method of displaying content on a display of a device”

232. The preamble of claim 12 of the ’755 recites, in part, “[a] method of displaying content on a display of a device.” It is my opinion that *Huang* discloses this aspect of element 12.pre for the same reasons discussed above. (*Supra* Section X.B.1.a.) As discussed above, *Huang*’s disclosures would have been understood to demonstrate that *Huang*’s player presents a user interface for content defined by *Huang*’s applications. (*Id.*)

- (ii) “utilizing a registry of one or more web components related to inputs and outputs of a web service obtainable over a network, where each web component includes a plurality of symbolic names of inputs and outputs associated with each web service and where the registry includes a) symbolic names required for evoking one or more web components each related to a set of inputs and outputs of a web service obtainable over a network”

233. In my opinion, *Huang* in view of *Shenfield* renders this aspect of element 12.pre obvious for the same reasons discussed above. (*Supra* Sections X.B.1.b.) To summarize, a POSITA would have been motivated to modify the system of *Huang* to implement a *registry* as disclosed in *Shenfield*, and the *registry* would contain *Huang*'s identifiers—*symbolic names* that are character strings of alphanumeric symbols—for its generic objects (including *symbolic names* for web objects, which are *web components*, and their attributes, which are the *inputs and outputs of a web service*), which are *required* to call up and *evoke the functionality of web services that are obtainable over the World Wide Web*, which is a *network*. (*Id.*)

- (iii) “where the symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service”

234. It is my opinion that *Huang* discloses this aspect of element 12.pre for the same reasons discussed above. (*Supra* Section X.B.1.b(iii).) As stated above, *Huang*'s names for its generic web object attributes are character strings that do not contain any address or resource location information and, accordingly, do not include a persistent address or a pointer to an output value accessible to the web service. (*Id.*)

- (iv) “[where the registry includes:] b) the address of the web service”

235. It is my opinion that *Huang* in view of *Shenfield* renders this aspect of element 12.pre obvious for the same reasons discussed above. (*Supra* Section X.B.1.b(iv).) To summarize, a POSITA would have been motivated to combine the *registry* of *Shenfield* with the system of *Huang* and the *registry* would include the *address of the web service* because this is information necessary to access a web service. (*Id.*)

b. [12.a] “defining a user interface (UI) object for presentation on the display, where said UI object corresponds to a web component included in said registry selected from the group consisting of an input of the web service and an output of the web service;”

236. It is my opinion that *Huang* in view of *Shenfield* renders element 12.a obvious for the same reasons discussed above. (*Supra* Section X.B.1.c.) To summarize, the *Huang-Shenfield* system would contain *Huang*’s designer, which allows for defining application components, which are *user interface (UI) objects for presentation on the display*, and the designer would establish bindings between these application components and the attributes (*inputs and outputs of a web service*) of web objects (*web components*) through symbolic names that are included in a *registry*. (*Id.*)

c. [12.b] “selecting a symbolic name from said web component corresponding to the defined UI object;”

237. It is my opinion that *Huang* discloses this element 12.b for the same reasons discussed above. (*Supra* Section X.B.1.d.) To summarize, *Huang* discloses

that its designer can be used to *select a symbolic name of a web component that corresponds to the defined UI object*. (*Id.*)

d. [12.c] “associating the selected symbolic name with the defined UI object;”

238. It is my opinion that *Huang* discloses element 12.c for the same reasons discussed above. (*Supra* Section X.B.1.e.) As discussed above, *Huang* discloses that its system allows to generate bindings that *associate the selected symbolic name of a web object attribute to the application component (defined UI object)*. (*Id.*)

e. [12.d] “producing an Application including the selected symbolic name of the defined UI object, where said Application is a device-dependent [*sic*] code; and”

239. I understand that this limitation contains an error. As discussed above, in my opinion, the phrase “is a device-dependent code” should be read as “is a device-independent code” in view of the specification and prosecution history. (*Supra* Section VII.) Specifically, this correction is not subject to reasonable dispute and the prosecution history does not contain any indication of an alternative construction. (*Id.*)

240. For purposes of this analysis, I have been asked to assume that this claim limitation reads “producing an Application including the selected symbolic name of the defined UI object, where said Application is a device-independent code.”

241. In my opinion, it is my opinion that *Huang* discloses element 12.d for the same reasons discussed above. (*Supra* Section X.B.1.f.) To summarize, *Huang* discloses *producing* XML applications (*application in a device-independent code*) that contain code that *includes the symbolic names* of the *selected* attribute that corresponds to the application component (*defined UI object*). (*Id.*)

f. [12.e] “producing a Player, where said Player is a device-dependent code;”

242. It is my opinion that *Huang* in view of *Shenfield* renders element 12.e obvious for the same reasons discussed above. (*Supra* Section X.B.1.g.) To summarize, the modified *Huang-Shenfield* system renders obvious *producing* device-specific runtime players (*a Player written in device-dependent code*).

g. [12.f] “such that, when the Application and Player are provided to the device and executed on the device, and when a user of the device provides one or more input values associated with an input symbolic name to an input of defined UI object, 1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service, 2) the web service utilizes the input symbolic name and the user provided one or more input values for generating one or more output values having an associated output symbolic name, 3) said Player receives the output symbolic name and corresponding one or more output values and provides instructions for a display of the device to present an output value in the defined UI object.”

243. It is my opinion that *Huang* in view of *Shenfield* renders element 12.f obvious for the same reasons discussed above. (*Supra* Section X.B.1.h.) As

discussed above, the modified *Huang-Shenfield* system renders obvious each aspect of this limitation.

8. Claim 14

- a. “The method of claim 12, where said web component is a text chat, a video chat, an image, a slideshow, a video, or an RSS feed.”**

244. Claim 14 is identical to claim 3, except that claim 14 is a method claim that depends on claim 12. In my opinion, claim 12 is obvious for the reasons stated above. (*Supra* Section X.B.7.) It is further my opinion that claim 14 is obvious over *Huang* in view of *Shenfield* for the same reasons that this rendered claim 3 obvious. (*Supra* Section X.B.2.)

9. Claim 16

- a. “The method of claim 12, where said UI object is an input field for a web service.”**

245. Claim 16 is identical to claim 5, except that claim 16 is a method claim that depends on claim 12. In my opinion, claim 12 is obvious for the reasons stated above. (*Supra* Section X.B.7.) It is further my opinion that claim 16 is obvious over *Huang* in view of *Shenfield* for the same reasons that this rendered claim 5 obvious. (*Supra* Section X.B.3.)

10. Claim 17

- a. **“The method of claim 12, where said UI object is an input field usable to obtain said web component, where said input field includes a text field, a scrolling text box, a check box, a drop down-menu, a list menu, or a submit button.”**

246. Claim 17 is identical to claim 6, except that claim 17 is a method claim that depends on claim 12. In my opinion, claim 12 is obvious for the reasons stated above. (*Supra* Section X.B.7.) It is further my opinion that claim 17 is obvious over *Huang* in view of *Shenfield* for the same reasons that this rendered claim 6 obvious. (*Supra* Section X.B.4.)

11. Claim 18

- a. **“The method of claim 12, where said web component is an output of a web service, is the text provided by one or more simultaneous chat sessions, is the video of a video chat session, is a video, an image, a slideshow, an RSS display, or an advertisement.”**

247. Claim 18 is identical to claim 7, except that claim 18 is a method claim that depends on claim 12. In my opinion, claim 12 is obvious for the reasons stated above. (*Supra* Section X.B.7.) It is further my opinion that claim 18 is obvious over *Huang* in view of *Shenfield* for the same reasons that this rendered claim 7 obvious. (*Supra* Section X.B.5.)

12. Claim 22

a. “The method of claim 12, further comprising: providing said Application and Player over said network.”

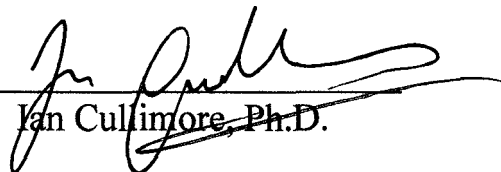
248. In my opinion, claim 22 is substantially similar to claim 11, except that it is a method claim that depends on claim 12. The primary difference in the language of the claim is that claim 22 recites “providing said Application and Player over said network,” whereas claim 11 recites “where said code is provided over said network.” (Ex-100, Claims 11, 22.)

249. In my opinion, the *Huang-Shenfield* system renders this limitation obvious. As discussed previously, the *Huang-Shenfield* system provides code over the network in the form of an Application and Player and, therefore, *provides the Application and Player over the network*. (*Supra* Section X.B.6.) Accordingly, claim 22 is obvious over *Huang* and *Shenfield* for the same reasons as claim 11. (*Id.*)

XI. CONCLUSION

250. I declare that all statements made herein of my knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code. I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Date: July 5th, 2021

By: 
Ian Cullimore, Ph.D.