

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

FACEBOOK, INC., ATCLASSIAN CORP. PLC, ATCLASSIAN, INC.,
EXPEDIA, INC., HOMEAWAY.COM, INC.,
PINTEREST, INC., SQUARESPACE, INC., WIX.COM, LTD.,
WIX.COM, INC., and GOOGLE LLC,¹
Petitioners,

v.

EXPRESS MOBILE INC.,
Patent Owner.

IPR2021-01456
Patent 9,471,287 B2

Before JEFFREY S. SMITH, AMBER L. HAGY, and
AARON W. MOORE, *Administrative Patent Judges*.

HAGY, *Administrative Patent Judge*.

JUDGMENT
Final Written Decision
Determining All Challenged Claims Unpatentable
35 U.S.C. § 318(a)

¹ Atlassian Corp. PLC., Atlassian, Inc., Expedia, Inc., Homeaway.com, Inc., Pinterest, Inc., Squarespace, Inc., Wix.com, Ltd., and Wix.com, Inc., filed a motion for joinder and a petition in Case IPR2022-00784. Google LLC filed a motion for joinder and a petition in Case IPR2022-00789. All of these parties have been joined as petitioners in IPR2021-01456.

I. INTRODUCTION

In this *inter partes* review, Facebook, Inc., Atlassian Corp. PLC., Atlassian, Inc., Expedia, Inc., Homeaway.com, Inc., Pinterest, Inc., Squarespace, Inc., Wix.com, Ltd., Wix.com, Inc., and Google LLC (collectively, “Petitioners”) challenge the patentability of claims 1, 2, 5–7, 11, and 12 (“the challenged claims”) of U.S. Patent No. 9,471,287 B2 (Ex. 1001, the “’287 patent”), which is assigned to Express Mobile, Inc. (“Patent Owner”).

We have jurisdiction under 35 U.S.C. § 6(b). For the reasons that follow, we determine that Petitioner has shown by a preponderance of the evidence that claims 1, 2, 5–7, 11, and 12 of the ’287 patent are unpatentable. *See* 35 U.S.C. § 316(e) (2018) (“In an *inter partes* review instituted under this chapter, the petitioner shall have the burden of proving a proposition of unpatentability by a preponderance of the evidence.”). This Final Written Decision is issued pursuant to 35 U.S.C. § 318(a).

A. Procedural History

Facebook, Inc. filed a Petition (Paper 1, “Pet.”) challenging claims 1, 2, 5–7, 11, and 12 of the ’287 patent on the following grounds (Pet. 6–7):

Claim(s) Challenged	35 U.S.C. §	Reference(s) / Basis
1, 2, 5–7, 12	103(a) ²	Anderson, ³ Bowers, ⁴ Jacobs, ⁵ Ambrose-Haynes, ⁶ and Geary ⁷
11	103(a)	Anderson, Bowers, Jacobs, Ambrose-Haynes, Geary, and NFS Administration ⁸

Patent Owner filed a Preliminary Response. Paper 6. A trial was instituted on the asserted grounds of unpatentability. Paper 10 (“Dec.”) 15.

During the trial, Patent Owner filed a Patent Owner Response. Paper 17 (“PO. Resp.”). Petitioner filed a Reply. Paper 18 (“Reply”). With authorization from the Board, Petitioner and Patent Owner filed additional briefing addressing Patent Owner’s alleged objective evidence of

² The Leahy-Smith America Invents Act (“AIA”), Pub. L. No. 112-29, 125 Stat. 284 (2011), amended 35 U.S.C. § 103, effective March 16, 2013. Because the ’287 patent has an effective filing date before this date, the pre-AIA version of § 103 applies.

³ G. Anderson & P. Anderson, *JAVA Studio Creator Field Guide*, 2d ed. (Prentice Hall 2006) (Ex. 1003, “Anderson”).

⁴ B. Bowers & S. Lane, *Advanced FileMaker Pro 6 Web Development* (Wordware Pub. 2003) (Ex. 1004, “Bowers”).

⁵ S. Jacobs, *Foundation XML for Flash* (Springer-Verlag 2006) (Ex. 1005, “Jacobs”).

⁶ N. Ambrose-Haynes et al., *Professional ColdFusion 5.0* (Wrox Press. Ltd. 2001) (Ex. 1006, “Ambrose-Haynes”).

⁷ D. Geary & C. Horstmann, *Core JavaServer Faces* (Sun Microsystems Press, Prentice Hall 2004) (Ex. 1011, “Geary”).

⁸ *Unix System V NFS Administration* (D. Herman, ed., Prentice Hall 1993) (Ex. 1007, “NFS Administration”).

nonobviousness. Paper 26 (“PO Obj. Evid. Br.”); Paper 27 (“Pet. Obj. Evid. Resp. Br.”).

After institution, Expedia, Inc., eBay Inc., Homeaway.com, Inc., Squarespace, Inc., Wix.com, Ltd., and Wix.com, Inc. filed a motion for joinder and a petition in IPR2022-00784. Google LLC filed a motion for joinder and a petition in IPR2022-00789. We joined all of these parties as petitioners in this proceeding. Papers 25, 37.

On December 6, 2022, we conducted an oral hearing. A copy of the transcript (Paper 36, “Tr.”) is included in the record.

Petitioner relies on testimony from Dr. Vijay K. Madisetti. Exs. 1002 (Declaration), 1021 (Reply Declaration). Patent Owner relies on testimony from Dr. Kevin C. Almeroth. Exs. 2001 (Declaration), 2020 (Second Declaration). A record transcript for the deposition of Dr. Madisetti has been entered into the record. Ex. 2025.

B. Real Parties in Interest

The following are the identified real parties in interest on the petitioner side of the proceeding, accounting for all of the joined parties:

Facebook, Inc., Atlassian Corp. PLC, Atlassian, Inc., Atlassian (UK) Limited, Atlassian (UK) Holdings Limited, Expedia, Inc., Homeaway.com, Inc., Expedia Group, Inc., VRBO Holdings, Inc., Pinterest, Inc., Squarespace, Inc., Wix.com, Ltd. and Wix.com, Inc. and Google LLC.
Pet. 1; IPR2022-00784, Paper 1 at 1; IPR2022-00789, Paper 1 at 1.

Patent Owner identifies itself as the real party in interest. Paper 4 at 1.

C. Related Matters

As required by 37 C.F.R. § 42.8(b)(2), the parties identify various related matters, including numerous district court proceedings involving the

'287 patent and/or its parent patent, U.S. Patent No. 9,063,755 (“the ’755 patent”). Pet. 1–4; Paper 4, 1–5; IPR2022-00784, Paper 1 at 1–5; IPR2022-00789, Paper 1 at 1–5. Among the identified related matters are several *inter partes* matters involving the ’287 patent: IPR2021-00710 (“the ’710 IPR”), IPR2021-01145 (“the ’1145 IPR”), IPR2021-01227 (“the ’1227 IPR”), and IPR2022-00248 (“the ’248 IPR”). Pet. 2; Paper 4, 2, 5.⁹

D. The ’287 Patent (Ex. 1001)

The ’287 patent relates to a platform for generating and distributing programming to mobile devices over a network. Ex. 1001, Abs. The system includes a database of web services that can be obtained over a network, and an authoring tool “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.* at 1:36–42. The Specification describes device-independent programs (“Applications”) and device- or platform-specific instructions (“Players”) that are provided to user devices. *Id.* at 5:8–14. A Player transforms device-independent instructions of an Application into device-specific instructions that are executable by a user device to, for example, generate one or more pages on the display of the device. *Id.* at 5:60–63, 6:4–8. The disclosed system allows for Applications

⁹ Petitioner misidentifies several IPR matters as involving the ’287 patent, which in fact involve different patents owned by Patent Owner. *See* Pet. 2 (citing IPR2021-00700 (involving U.S. Patent No. 6,546,397); IPR2021-01144 (involving the ’755 patent); and IPR2021-01228 (involving the ’755 patent)). We view this misidentification as inadvertent error.

to provide web service interaction and invocation to a device. *Id.* at 7:47–56.

Figure 2A illustrates the interaction between system components, and is reproduced below.

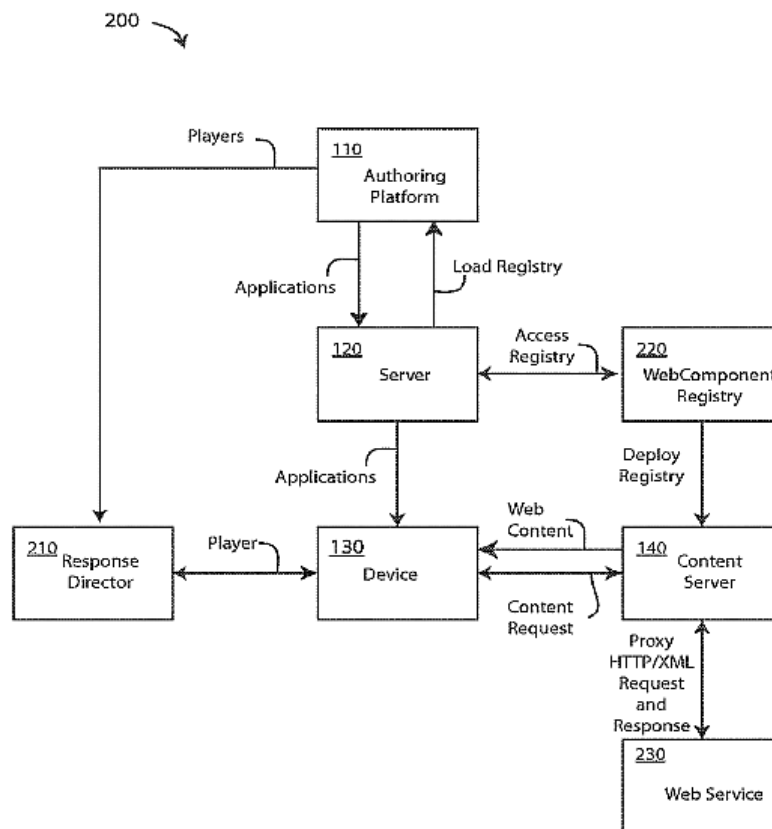


FIG. 2A

Figure 2A is a schematic illustrating the communications between different system components. Ex. 1001, 2:21–23.

As illustrated in Figure 2A, authoring platform 110 may generate one or more Players, which are provided to response director 210. *Id.* at 8:15–17. Device 130 requests a Player from response director 210, and receives and installs the Player. *Id.* at 8:18–20. Web service 230 includes a plurality of services obtainable over the Internet. *Id.* at 8:26–27. Each web service is identified in an entry in web component registry 220. *Id.* at 8:27–30. Web

component registry 220 is provided through server 120 to authoring platform 110 so that a user of the authoring platform may bind web services 230 to elements to be displayed on device 130. *Id.* at 8:30–34. A web component registry 220 for each registered web service 230 is loaded into authoring platform 110. *Id.* at 8:62–64. The user of the authoring platform can then assign components of any web service 230 to an Application without any need to write code. *Id.* at 8:64–66.

E. Illustrative Claim

Of the challenged claims, claim 1 is independent. Challenged claims 2, 5–7, 11, and 12 depend from claim 1. Claim 1, reproduced below, illustrates the claimed subject matter:

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, where each symbolic name has an associated data format class type corresponding to a subclass of User Interface (UI) objects that support the data format type of the symbolic name, and has a preferred UI object, and
- b) an address of the web service;

an authoring tool configured to:

define a (UI) object for presentation on the display, where said defined UI object corresponds to a web component included in said registry selected from a group consisting of an input of the web service and an output of the web service, where each defined UI object is

- either: 1) selected by a user of the authoring tool; or
2) automatically selected by the system as the preferred
UI object corresponding to the symbolic name of the
web component selected by the user of the authoring
tool,
- 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, where the selected symbolic name is only
available to UI objects that support the defined data
format associated with that symbolic name, and
- produce an Application including the selected symbolic
name of the defined UI object, where said Application
is a device-independent code; and
- a Player, where said Player is a device-dependent code,
wherein, when the Application and Player are provided
to the device and executed on the device, and when the
user of the device provides one or more input values
associated with an input symbolic name to an input of
the 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 the display of the device to present an
output value in the defined UI object.

Ex. 1001, 37:48–38:37.

II. ANALYSIS

A. *Principles of Law*

A patent claim is unpatentable under 35 U.S.C. § 103(a) if the differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, “would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.” *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). The question of obviousness is resolved on the basis of underlying factual determinations including: (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of ordinary skill in the art; and (4) when in evidence, objective evidence of nonobviousness. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966).

“In an [*inter partes* review], the petitioner has the burden from the onset to show with particularity why the patent it challenges is unpatentable.” *Harmonic*, 815 F.3d at 1363 (citing 35 U.S.C. § 312(a)(3) (requiring *inter partes* review petitions to identify “with particularity . . . the evidence that supports the grounds for the challenge to each claim”)). This burden of persuasion never shifts to Patent Owner. *See Dynamic Drinkware, LLC v. Nat’l Graphics, Inc.*, 800 F.3d 1375, 1378 (Fed. Cir. 2015) (discussing the burden of proof in *inter partes* review).

B. *Level of Ordinary Skill in the Art*

The level of skill in the art is a factual determination that provides a primary guarantee of objectivity in an obviousness analysis. *Al-Site Corp. v. VSI Int’l Inc.*, 174 F.3d 1308, 1324 (Fed. Cir. 1999) (citing *Graham*, 383

U.S. at 17–18; *Ryko Mfg. Co. v. Nu-Star, Inc.*, 950 F.2d 714, 718 (Fed. Cir. 1991)).

Relying on the declaration testimony of Dr. Madisetti, Petitioner asserts:

A person of ordinary skill as of April 2008 would have had at least a bachelor’s degree in software engineering, computer science, computer engineering, or electrical engineering with at least two years of experience in web-based software application development, including experience in developing software and systems for storing, retrieving, and transmitting information (such as text and images) over a computer network such as the Internet (or equivalent degree or experience).

Pet. 8 (citing Ex. 1002 ¶ 16).

Patent Owner does not propose an alternative assessment. *See generally* PO Resp. We note that Patent Owner’s declarant, Dr. Almeroth, describes the level of ordinary skill in the art similarly to the assessment offered by Petitioner. *See* Ex. 2001 ¶ 31.

Having reviewed the arguments and evidence in the full record, we adopt Petitioner’s definition above, as we did initially in the Institution Decision, as it is consistent with the ’287 patent and the asserted prior art.

C. Claim Construction

In interpreting the claims of the ’287 patent, we “us[e] the same claim construction standard that would be used to construe the claim[s] in a civil action under 35 U.S.C. [§] 282(b).” *See* 37 C.F.R. § 42.100(b) (2021). The claim construction standard includes construing claims in accordance with the ordinary and customary meaning of such claims as would have been understood by one of ordinary skill in the art in light of the written

description and the prosecution history pertaining to the patent. *See id.*; *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312–14 (Fed. Cir. 2005) (en banc).

Neither party proposes any terms for construction. Petitioner notes some constructions of certain terms that were proposed by the parties in underlying litigation involving the '287 patent,¹⁰ but states that “Petitioner does not believe express claim construction is necessary at this time.” Pet. 10–11.

Patent Owner also does not propose any constructions, but asserts that Petitioner’s arguments are inconsistent with the construction of “device-independent,” which Patent Owner states was “proposed by Petitioner” in the district court “and agreed to by the parties.” PO Resp. 2 (emphasis omitted). In particular, Patent Owner states that “the term ‘device-independent code’ has been construed by a district court to mean ‘code that is not specific to the operating system, programming language, or platform of a device.’” *Id.* at 12 (citing Ex. 2021, 15–16; Ex. 2022, 10). Petitioner does not challenge this construction or offer a competing construction, but does disagree with Patent Owner’s application of this construction, as discussed further herein. *See* Reply 2–12; *see also infra* § II.E.1.a.(3)(e).

As the parties do not disagree with the District Court’s construction, we adopt it for use in this proceeding. We address Petitioner’s arguments

¹⁰ The parties include Patent Owner and several defendants (including Facebook, Expedia, eBay, Google, and Atlassian). *See* Ex. 2009. The underlying district court litigation was originally filed in the Western District of Texas, and that is where the parties submitted their claim construction briefing. *See id.* Before the court entered a final claim construction order, however, the case was transferred to the Northern District of California. *See* Prelim. Resp. 15, n.1; Paper 4, 2.

about the application of the construction in the course of the validity analysis. *See infra* § II.E.1.a.(3)(e).

Patent Owner’s arguments with regard to the “Player” limitation in claim 1 also highlight another potential claim construction dispute regarding the claim limitation “produce a player” as recited in claim 1. *See* PO Resp. 20–26. In particular, as Patent Owner points out, the parties in the underlying litigation disputed whether the recited “Player” in claim 1 is “something that is produced by the ‘authoring tool’ (like the preceding ‘Application’ limitation), or whether the ‘Player’ is completely separate from and unrelated to the claimed ‘authoring tool.’” Ex. 2029 (Defendants’ Responsive Claim Construction Brief), 26. The defendants in that litigation (including Petitioner) jointly proposed that claim 1 be construed to require that the “authoring tool” that is configured to carry out certain recited actions must also “produce” the claimed Player. *Id.* Patent Owner asserted that no construction is necessary. *Id.*

Although Patent Owner highlights this dispute and characterizes Petitioner’s position here as potentially inconsistent with its position in the district court, Patent Owner does not propose any construction. *See* PO Resp. 20–26. Petitioner notes the parties’ proposed constructions of several terms from an underlying litigation, as noted above, but does not mention this particular dispute. *See* Pet. 10–11. Petitioner states that no express construction of any terms is needed (Pet. 10), and thus does not advocate, in this forum, for construing “a Player” as produced by the authoring tool.

Although we lacked guidance from the parties on this issue at the institution stage, we nevertheless considered whether claim 1 requires that the Player be produced by the authoring tool, or whether the Player is simply

part of the recited system for generating code (which, of course, would not exclude its being produced by the authoring tool). *See* Dec. Inst. 17–22. We concluded, that, on its face, the recitation of “a Player” as part of a system claim does not suggest the need for indicating the origin of the Player. We determined that construing “a Player” as something that is specifically *produced by the authoring tool* would require us effectively to *add language* to claim 1 (inserting “produce” before “a Player,” to tie back to “an authoring tool configured to”). *Id.* at 20.

We then explained that “[t]his does not strike us as a matter of claim construction but rather as a matter of correction to the claim.” *Id.* Although we noted that “district courts may correct obvious errors in patent claims by interpretation of the patent,” we concluded on the record then before us, and especially in the absence of guidance by the parties, that the situation did not present an “obvious and correctable error in the claim” so as to warrant correction. *Id.* (citing *CBT Flint Partners, LLC v. Return Path, Inc.*, 654 F.3d 1353, 1358 (Fed. Cir. 2011) (holding that where “there is an obvious and correctable error in the claim, the construction of which is not subject to reasonable debate,” correction is proper); *Novo Indus., L.P. v. Micro Molds Corp.*, 350 F.3d 1348, 1354 (Fed. Cir. 2003) (setting out two-part test for correction where “correction is not subject to reasonable debate”)).

We then turned to claim 1 itself, and observed that, on the one hand, the formatting structure of claim 1 places “a Player” at the same indentation level as several other limitations that appear to be sub-limitations of “an authoring tool configured to.” Dec. 21. We noted that Petitioner made the same observation before the district court, and argued that “the claim as structured makes no sense if no verb precedes ‘a Player.’” *Id.* (citing

Ex. 2009, 27). We then observed that, on the other hand, the punctuation of the claim gives a different impression. In particular, the action-verb sub-limitations following “an authoring tool configured to” are each separated by a *comma*—whereas the recitation of “a Player” is separated from the previous limitation by a *semicolon*. *Id.* In the structure of claim 1, this punctuation suggests that “a Player” is not a sub-limitation of “an authoring tool configured to,” but is instead a component *at the same level* as an authoring tool. *Id.* We based this observation on the consistent use of commas in claim 1 to differentiate sub-limitations, as opposed to the consistent use of semicolons to separate limitations at a higher level. *Id.* We found this differentiation in punctuation to be a more compelling indication of the hierarchy of the limitations than the relative indentation of the clauses. *Id.*

Ultimately, in the Institution Decision, we determined that, on its face, claim 1 is subject to a reasonable reading in which no action verb must precede “a Player.” *Id.* at 22. We therefore declined to interpret “a Player,” as recited in claim 1, as something that the authoring tool must “produce,” and stated that no further construction is necessary. *Id.*

Now, at the conclusion of the trial in this matter, we are still left without additional guidance from the parties. Accordingly, having now considered the full record at trial, and in the absence of any persuasive arguments by the parties, we follow the same reasoning as we did in the Institution Decision. We decline to interpret “a Player,” as recited in claim 1, as something that the authoring tool must “produce.” No further construction is necessary.

We determine that no explicit construction of any other terms is needed to resolve the issues presented by the arguments and evidence of record. *See Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017) (per curiam) (claim terms need to be construed “only to the extent necessary to resolve the controversy” (quoting *Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999))).

D. Overview of Cited References

1. Anderson (Ex. 1003)

Anderson is an excerpt from a book published in 2006. Ex. 1003, 4.¹¹ Petitioner asserts that Anderson is prior art because it was published before April 2008, as shown by its “public accessibility and indexing in public libraries.” Pet. 15. Petitioner also relies on the Declaration of Sylvia Hall-Ellis, Ph.D., who provides additional details as to the publication and public availability of Anderson. Ex. 1015 ¶¶ 43–49.

We evaluated the evidence submitted by Petitioner at the institution stage and determined that Petitioner had shown sufficiently for purposes of institution that Anderson qualifies as a printed publication. *See* Inst. Dec. 23–24 (citing *Hulu, LLC v. Sound View Innovations, LLC*, IPR2018-01039, Paper 29 at 16 (PTAB Dec. 20, 2019) (precedential)). Patent Owner does not challenge Petitioner’s showing or our initial determination. *See generally* PO Resp.

¹¹ References to page numbers for this exhibit are to the page numbers as provided by Petitioner on the exhibit.

We determine, based on the full record of the proceeding, that Petitioner has submitted sufficient evidence to establish that Anderson was publicly accessible before the critical date of the challenged patent, and, thus Petitioner has established that the reference qualifies as a printed publication. In particular, the face of Anderson bears a copyright date of 2006, a publication date of May 2006, an ISBN (“International Standard Book Number”), and identifies an established publisher (Prentice Hall). Ex. 1003, 4. *See Hulu* at 19 (noting that Petitioner had met its burden sufficiently for institution by submitting a copy of a reference that bore “a copyright date of 1990, a printing date of November 1992, and an ISBN date of 8/94,” and was part of “a textbook from an established publisher . . . and a well-known book series”). Dr. Hall-Ellis testifies that Exhibit 1003 is a true and correct copy of the selected pages from the book as held by the Library of Congress, and also attests that the book was publicly available by or shortly after June 28, 2007, because by that date it had been received, cataloged, and indexed at the Library of Congress and made part of the Online Computer Library Center’s (“OCLC’s”) bibliographic database. Ex. 1015 ¶¶ 43, 49; *see id.* ¶ 28 (describing OCLC).

Anderson is an excerpt from a book describing a software development tool known as Java Studio Creator (or “Creator”), which developers may use to create web applications. Ex. 1003, 35. For example, Anderson describes a Java application (Google1) that allows users to enter a search query and then access a Google web service (using a method called doGoogleSearch) to perform a search based on that query. *Id.* at 285, 289. Anderson also describes how to design the user interface for such application, including incorporating the Google logo, a text field entry box,

and a search button, and to interface with Google's servers to utilize Google web services to perform the search and display the results. *Id.* at 286–288.

2. *Bowers (Ex. 1004)*

Bowers is an excerpt from a book titled “Advanced FileMaker® Pro 6 Web Development,” published in 2003. Ex. 1004, 3.¹² Petitioner asserts that Bowers is prior art because it was published before April 2008, as shown by its “public accessibility and indexing in public libraries.” Pet. 15. Petitioner also relies on the Declaration of Dr. Hall-Ellis, who provides additional details as to the publication and public availability of Bowers. Ex. 1015 ¶¶ 50–57.

We evaluated the evidence submitted by Petitioner at the institution stage and determined that Petitioner had shown sufficiently for purposes of institution that Bowers qualifies as a printed publication. *See* Dec. 25–26. Patent Owner does not challenge Petitioner's showing or our initial determination. *See generally* PO Resp.

We determine, based on the full record of the proceeding, that Petitioner has submitted sufficient evidence to establish that Bowers was publicly accessible before the critical date of the challenged patent, and, thus Petitioner has established that the reference qualifies as a printed publication. In particular, the face of Bowers bears a copyright date of 2003, a publication date of 2003, and an ISBN, and identifies an established publisher (Wordware Publishing). Ex. 1004, 3. *See Hulu* at 19. Dr. Hall-Ellis testifies that Exhibit 1004 is a true and correct copy of the selected

¹² References to page numbers for this exhibit are to the page numbers as provided by Petitioner on the exhibit.

pages from the book as held by the Library of Congress, and also attests that the book was publicly available by or shortly after April 19, 2004, because by that date it had been received, cataloged, and indexed at the Library of Congress and made part of the OCLC bibliographic database. Ex. 1015 ¶¶ 50, 56.

Bowers describes techniques for developing web applications using FileMaker Pro, a commercially-available database software program. Ex. 1004, 68–121. Bowers describes the GoogleSearch web service—the same service discussed in Anderson—as a concrete example of how to incorporate a web service into an application. *Id.* at 97–110.

3. *Jacobs (Ex. 1005)*

Jacobs is an excerpt from a book titled “Foundation XML for Flash,” published in 2006. Ex. 1005, 3.¹³ Petitioner asserts that Jacobs is prior art because it was published before April 2008, as shown by its “public accessibility and indexing in public libraries.” Pet. 15. Petitioner also relies on the Declaration of Dr. Hall-Ellis, who provides additional details as to the publication and public availability of Jacobs. Ex. 1015 ¶¶ 58–65.

We evaluated the evidence submitted by Petitioner at the institution stage and determined that Petitioner had shown sufficiently for purposes of institution that Jacobs qualifies as a printed publication. *See* Dec. 26–27. Patent Owner does not challenge Petitioner’s showing or our initial determination. *See generally* PO Resp.

¹³ References to page numbers for this exhibit are to the page numbers as provided by Petitioner on the exhibit.

We determine, based on the full record of the proceeding, that Petitioner has submitted sufficient evidence to establish that Jacobs was publicly accessible before the critical date of the challenged patent, and, thus Petitioner has established that the reference qualifies as a printed publication. In particular, the face of Jacobs bears a copyright date of 2006, an ISBN, and identifies an established publisher (Springer-Verlag). Ex. 1005, 3. *See Hulu* at 19. Dr. Hall-Ellis testifies that Exhibit 1005 is a true and correct copy of the selected pages from the book as held by the Library of Congress, and also attests that the book was publicly accessible shortly after November 15, 2005, when it was received, catalogued, and indexed in the Langsam Library at the University of Cincinnati. Ex. 1015 ¶¶ 58–65.

Jacobs describes techniques for developing web applications using a technology known as Flash. Ex. 1005, 16. Like Anderson and Bowers, Jacobs includes a discussion of how to incorporate web services into a web application. *Id.* at 101–106. Jacobs explains that an industry standard technology known as the Simple Object Access Protocol (“SOAP”) is used to specify the messages exchanged between a web service and a third party application that uses the web service. *Id.* at 65, 102. Jacobs shows an example of messages exchanged using SOAP in connection with the GoogleSearch web service, and in particular, to run a search. *Id.* at 102–106.

4. *Ambrose-Haynes (Ex. 1006)*

Ambrose-Haynes is an excerpt from a book titled “Professional ColdFusion 5.0,” published in 2001. Ex. 1006, 6.¹⁴ Petitioner asserts that Ambrose-Haynes is prior art because it was published before April 2008, as shown by its “public accessibility and indexing in public libraries.” Pet. 15. Petitioner also relies on the Declaration of Dr. Hall-Ellis, who provides additional details as to the publication and public availability of Ambrose-Haynes. Ex. 1015 ¶¶ 66–73.

We evaluated the evidence submitted by Petitioner at the institution stage and determined that Petitioner had shown sufficiently for purposes of institution that Ambrose-Haynes qualifies as a printed publication. *See* Dec. 27–28. Patent Owner does not challenge Petitioner’s showing or our initial determination. *See generally* PO Resp.

We determine, based on the full record of the proceeding, that Petitioner has submitted sufficient evidence to establish that Ambrose-Haynes was publicly accessible before the critical date of the challenged patent, and, thus Petitioner has established that the reference qualifies as a printed publication. In particular, the face of Ambrose-Haynes bears a copyright date of 2001 and an ISBN. Ex. 1006, 6. *See Hulu* at 19. Dr. Hall-Ellis testifies that Exhibit 1006 is a true and correct copy of the selected pages from the book as held by the Library of Congress, and also attests that the book was publicly available by or shortly after August 2, 2001, when it

¹⁴ References to page numbers for this exhibit are to the page numbers as provided by Petitioner on the exhibit.

was received, catalogued, and indexed in the Multnomah County Library in Portland, Oregon. Ex. 1015 ¶¶ 66–72.

Ambrose-Haynes describes a Java Virtual Machine (“JVM”), which is responsible for carrying out execution of Java applications. Ex. 1006, 55. Ambrose-Haynes explains that “JVMs are written for each operating platform”; therefore, “[e]xecuting the code through a JVM solves the problem of portability, since JVMs are written for each operating platform. In this way it is the JVM which is linked to the operating system, and handles all of the platform-specific details, while the source code [written] for interpretation is platform independent.” *Id.*

5. *NFS Administration (Ex. 1007)*

NFS Administration comprises selections from a book titled “UNIX® System V NFS Administration.” Ex. 1007, 1–3.¹⁵ Petitioner asserts that NFS Administration is prior art because it was published before April 2008, as shown by its “public accessibility and indexing in public libraries.” Pet. 15. Petitioner also relies on the Declaration of Dr. Hall-Ellis, who provides additional details as to the publication and public availability of NFS Administration. Ex. 1015 ¶¶ 74–81.

We evaluated the evidence submitted by Petitioner at the institution stage and determined that Petitioner had shown sufficiently for purposes of institution that NFS Administration qualifies as a printed publication. *See* Dec. 29–30. Patent Owner does not challenge Petitioner’s showing or our initial determination. *See generally* PO Resp.

¹⁵ References to page numbers for this exhibit are to the page numbers as provided by Petitioner on the exhibit.

We determine, based on the full record of the proceeding, that Petitioner has submitted sufficient evidence to establish that NFS Administration was publicly accessible before the critical date of the challenged patent, and, thus Petitioner has established that the reference qualifies as a printed publication. In particular, the face of NFS Administration bears a copyright date of 1993, Library of Congress publication data, including an ISBN, and identifies an established publisher (Prentice Hall). Ex. 1007, 3. *See Hulu* at 19. Dr. Hall-Ellis testifies that Exhibit 1007 is a true and correct copy of the selected pages from the book as held by the Library of Congress, and also attests that the book was publicly available by or shortly after November 23, 1992, because by that date it had been received, cataloged, and indexed in the Library of Congress and made a part of the OCLC bibliographic database. Ex. 1015 ¶¶ 74–80.

NFS Administration describes a technology known as the Network File System (“NFS”). Ex. 1007, 10. NFS provides for a distributed file system that allows a computer to access files from remote storage over a computer network, such as a server connected to the Internet. *Id.* “NFS enables machines of different architectures running different operating systems to share resources across a network.” *Id.*

6. *Geary (Ex. 1011)*

Geary is an excerpt from a book titled “Core JavaServer Faces,” published in 2004. Ex. 1011, 3.¹⁶ Petitioner asserts that Geary is prior art because it was published before April 2008, as shown by its “public

¹⁶ References to page numbers for this exhibit are to the page numbers as provided by Petitioner on the exhibit.

accessibility and indexing in public libraries.” Pet. 15. Petitioner also relies on the Declaration of Dr. Hall-Ellis, who provides additional details as to the publication and public availability of Geary. Ex. 1015 ¶¶ 82–89.

We evaluated the evidence submitted by Petitioner at the institution stage and determined that Petitioner had shown sufficiently for purposes of institution that Geary qualifies as a printed publication. *See* Dec. 30–31. Patent Owner does not challenge Petitioner’s showing or our initial determination. *See generally* PO Resp.

We determine, based on the full record of the proceeding, that Petitioner has submitted sufficient evidence to establish that Geary was publicly accessible before the critical date of the challenged patent, and, thus Petitioner has established that the reference qualifies as a printed publication. In particular, the face of Geary bears a copyright date of 2004 and an ISBN, and identifies an established publisher (Prentice Hall). Ex. 1011, 3. *See Hulu* at 19. Dr. Hall-Ellis testifies that Exhibit 1011 is a true and correct copy of the selected pages from the book as held by the Library of Congress, and also attests that the book was publicly available by or shortly after October 20, 2004, because by that date it had been received, catalogued, and indexed in The Chinese University of Hong Kong and made a part of the OCLC bibliographic database. Ex. 1015 ¶¶ 82, 88.

Geary describes a technology known as JavaServer Faces (“JSF”). Ex. 1011, 1. Geary explains that JSF provides a specification for building and incorporating user interface components into web applications using Java, including text fields, checkboxes, and buttons. *Id.* at 81. Geary also discloses standard Java classes for building user interfaces for web

applications, including UIOutput and UIInput which, respectively, handle output to and input from the user. *Id.* at 29.

E. Analysis

1. Ground 1: Obviousness of Claims 1, 2, 5–7, and 12 over Anderson, Bowers, Jacobs, Ambrose-Haynes, and Geary

Petitioner contends that claims 1, 2, 5–7, and 12 are unpatentable under 35 U.S.C. § 103(a) over the combined teachings of Anderson, Bowers, Jacobs, Ambrose-Haynes, and Geary. Pet. 15–81.

Patent Owner does not address Petitioner’s arguments on a claim-by-claim basis, but challenges Petitioner’s showing across all claims on several bases: (1) the proposed combination fails to teach an application that is a device-independent code (PO Resp. 12–16); (2) the proposed combination fails to teach an authoring tool configured to produce a player (*id.* at 16–26); (3) the Petition fails to demonstrate that it would have been obvious to combine Anderson with Ambrose-Haynes in a client-side framework (*id.* at 26–34); and (4) the Petition fails to demonstrate that it would not have been obvious to combine “five different references” (*id.* at 34–47). Patent Owner also argues that objective indicia of nonobviousness, in the form of a jury verdict awarding damages for infringement (by an unrelated party) and rejecting a validity challenge (based on different references) in district court litigation, weighs against finding unpatentability here. PO Obj. Evid. Br. 2–5.

We address Patent Owner’s arguments herein in the context of evaluating Petitioner’s showing as to each claim element.

a. Claim 1

(1) Preamble: “[a] system for generating code to provide content on a display of a device”

Petitioner contends Anderson discloses the subject matter of the preamble by disclosing Java Studio Creator, a software development and authoring tool for building web applications. Pet. 15 (citing Ex. 1003, 35). Petitioner asserts that the claimed “code” reads on either (1) Java source code generated by the developer for the web application using Creator, or (2) compiled Java “bytecode” that, when executed, implements the web application. *Id.*

Petitioner also contends that Anderson discloses generating code “to provide content on a display of a device,” as recited. *Id.* at 16. In particular, Petitioner contends that “[a] key feature of Java is the ability to access web services provided by third parties, such as Google.” *Id.* Such web services include software application programming interfaces (“APIs”) that are accessible over a network in a heterogeneous (cross-platform) environment. *Id.* (citing Ex. 1003, 285); *see also* Ex. 1003, 50, 52.

In connection with claim 1, Petitioner generally relies upon the teachings of Anderson, namely a specific example of “how to create an application that uses the Google Web Service API,” which is used for receiving and displaying search results in response to a user request. Pet. 16. Petitioner states that Anderson describes how to design the user interface for the application, and provides exemplary code (written in Java) for accessing Google web services to receive and display search results based on a user-provided search query. *Id.* Anderson then shows how such an application provides content on the display of a device, e.g., search results provided

through the Google search web services. *Id.* at 16–17 (citing Ex. 1003, 307, Fig. 10-9).

Petitioner’s arguments and evidence as to Anderson are persuasive. Patent Owner does not present evidence or arguments particularly addressing Petitioner’s showing as to the preamble.

We are persuaded Petitioner has demonstrated that Anderson teaches “[a] system for generating code to provide content on a display of a device,” as recited in the preamble of claim 1.¹⁷

(2) “*computer memory storing a registry of:*”

Claim 1 also recites “computer memory storing a registry of . . . a) symbolic names . . . and b) an address of the web service.” Ex. 1001, 37:50–62.

Petitioner contends Anderson teaches or suggests a registry stored in computer memory, arguing that the claimed “registry” corresponds in Anderson to an XML file—namely, a Web Services Description Language (“WSDL”) file describing Google web services. Pet. 18. Petitioner further contends that the claimed “computer memory” corresponds in Anderson to the memory that stores the WSDL file. *Id.* at 17–18. Petitioner asserts that this memory may be on Google’s servers that store the GoogleSearch.wsdl file (*id.* at 18 (citing Ex. 1002 ¶ 78)); Petitioner also asserts that “it would also have been obvious that the WSDL file could have been stored in the memory of the developer computer running Creator.” *Id.* at 19 (citing Ex. 1002 ¶ 79). Petitioner reasons that “Creator as disclosed in Anderson

¹⁷ Neither party has taken a position on whether the preamble is limiting. As Petitioner persuasively shows that Anderson teaches the subject matter of the preamble, we need not determine whether the preamble is limiting.

allows the developer to access WSDL information from local storage instead of a remote server.” *Id.* Petitioner additionally relies on Bowers to confirm that the WSDL file described in Anderson is an XML file. *Id.* at 19–20 (citing Ex. 1004, 98–102). Because an XML file is within the Specification’s description of a “registry” (Ex. 1001, 8:26–30), Petitioner contends that the GoogleSearch.wsdl file described in both Anderson and Bowers is a “registry” as recited in claim 1. Pet. 20.

Petitioner’s arguments and evidence are persuasive. Patent Owner does not particularly challenge Petitioner’s showing as to this limitation.

Based on Petitioner’s persuasive argument and evidence, we find that Anderson (alone or combined with Bowers) teaches a “computer memory storing a registry,” as recited in claim 1. We address in the next two sections the particular components recited to be stored in the registry.

*(a) “symbolic names required for evoking
one or more web components . . .”*

Claim 1 recites that the registry stored in computer memory contains

- 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 symbolic names are character strings that do not contain either a persistent address or pointer to an output value accessible to the web service[.]

Ex. 1001, 37:51–61.

Petitioner contends that the ’287 patent specification does not define “symbolic name,” but states that, “in the context of computer programming, this term generally refers to a name (usually in the form of an alphabetical character string) used to identify an entity or object in a computer system, such as a particular process, function, device, or data.” Pet. 22; Ex. 1002

¶ 87 (citing IBM Dictionary of Computing (1994) (Ex. 1008), 5 (“**symbolic name** (1) In a programming language, a unique name used to represent an entity such as a field, file, data structure, or label.”)). Petitioner further contends that the claimed “symbolic names” correspond to at least the following names contained in the GoogleSearch.wsdl file (corresponding to the claimed “registry”) as described in Anderson: doGoogleSearch, doSpellingSuggestion, and doGetCachedPage. Pet. 22 (citing Ex. 1004, 105 (“The WSDL file tells us that Google supports messages such as doGoogleSearch() and doSpellingSuggestion().”)). Petitioner asserts that each of these symbolic names corresponds to a particular function or method (corresponding to the claimed “one or more web components”) provided by the GoogleSearch web service. *Id.* For example, doGoogleSearch invokes the Google web service to perform a web search based on a search query. *Id.* According to Petitioner, these symbolic names are also “required for evoking one or more web components,” as claimed, because the web application must use them to identify the particular method of the Google web services being requested. *Id.* at 23. The GoogleSearch.wsdl file (which Petitioner maps to the claimed “registry”) uses these names to uniquely identify these methods (e.g. <message name=“doGoogleSearch”>), which in turn identify the corresponding feature of the Google web service available to a web application. *Id.* at 24 (citing Ex. 1004, 102).

Petitioner further contends that the symbolic names described in Anderson are also “each related to a set of inputs and outputs of a web service obtainable over a network,” as also recited in claim 1. *Id.* To elaborate on this point, Petitioner points to the more detailed description of doGoogleSearch from Bowers, which identifies an “input message” and an

“output message” type for each of the three methods provided by the Google web service (doGetCachedPage, doSpellingSuggestion, and doGoogleSearch). Pet. 24–25 (citing Ex. 1004, 100, 102); Ex. 1002 ¶ 97.

Petitioner further contends that the Google web service is “obtainable over a network,” as also required by claim 1. Pet. 31 (emphasis omitted). In particular, Petitioner points to Bowers, which explains that the GoogleSearch.wsdl file “contains a <service> element that ‘points to a URL. This URL is the address of the actual web service . . .’” *Id.* (citing Ex. 1004, 102). Petitioner explains that the “<service> URL identified in the version of GoogleSearch.wsdl in Bowers is <http://api.google.com/search/beta2>.” *Id.* (citing Ex. 1004, 101). Petitioner reasons that “[a] web service available from a URL that points to ‘google.com’ is ‘obtainable over a network,’” as claimed. *Id.* (emphasis omitted). Petitioner additionally notes that the symbolic names described in Anderson and Bowers (e.g., “doGoogleSearch”) are character strings that “merely provide labels for the available methods; they do not contain a ‘persistent address’ or a ‘pointer,’ and do not by themselves identify the location (for storage or otherwise) of the web service or any output returned from it.” *Id.* at 31–32.

Petitioner’s arguments are persuasive and supported by the cited evidentiary disclosures. Patent Owner does not challenge Petitioner’s showing as to this particular limitation.

Based on Petitioner’s persuasive argument and evidence, we find that the combination of Anderson and Bowers teaches or suggests

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 symbolic names are

character strings that do not contain either a persistent address or pointer to an output value accessible to the web service[,]
as recited in claim 1.

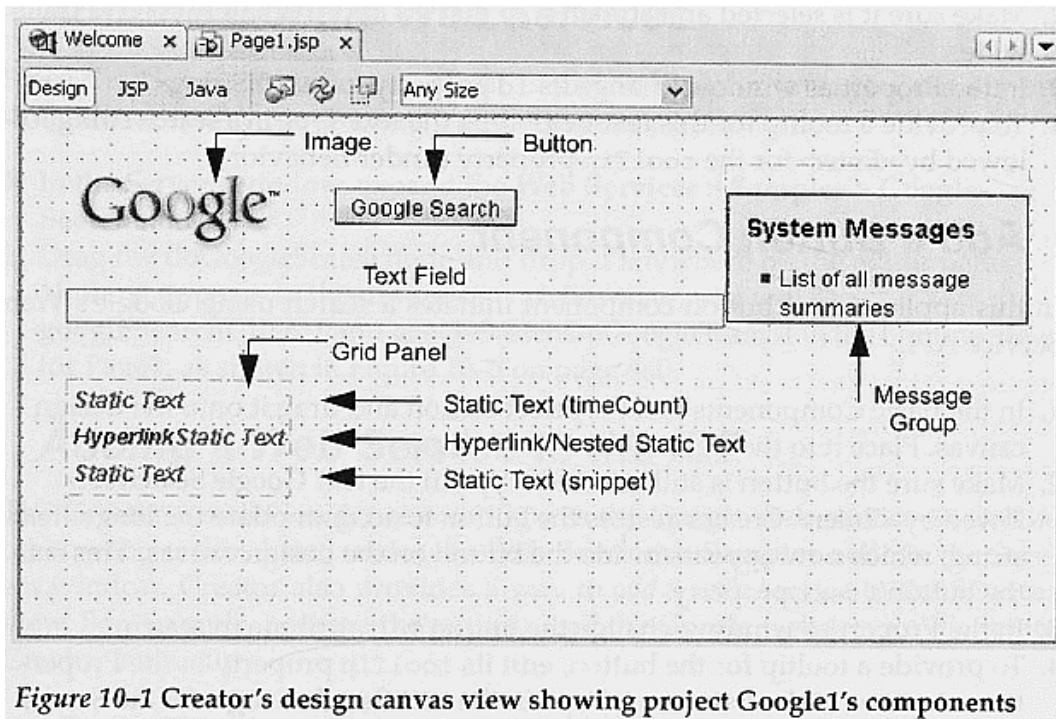
(b) “*each symbolic name has an associated data format class . . .*”

Claim 1 also recites, with regard to the symbolic names, that “each symbolic name has an associated data format class type corresponding to a subclass of User Interface (UI) objects that support the data format type of the symbolic name, and has a preferred UI object.” Ex. 1001, 37:57–61.

Petitioner contends, “[t]his limitation does little more than recite characteristics built into object-oriented programming systems such as Java – the end result of building the exemplary Java web application in Anderson for accessing the GoogleSearch web service.” Pet. 32. Relying in part on the expert testimony of Dr. Madisetti, Petitioner then describes Java’s object-oriented programming system, which is the system also described in Anderson and Bowers, in which procedures, known as “methods,” can be accessed (“called”) by other software. *Id.* at 32–33 (citing Ex. 1002 ¶ 101). In such an object-oriented system, Petitioner continues, the term “class” generally refers to the template for an object, and the term “subclass” is “simply a class that inherits [information and methods] from its parent class (referred to as a superclass).” *Id.* at 33 (citing Ex. 1002 ¶ 102; Ex. 1003, 43).

Petitioner then explains that this limitation reads on Anderson’s description of an exemplary Java web application that accesses the “doGoogleSearch” method of the Google web service. *Id.* at 34–35. In particular, Petitioner explains that the claimed “symbolic name” corresponds to the symbolic name from the GoogleSearch.wsdl file, e.g.,

“doGoogleSearch.” *Id.* at 35. Petitioner also explains that this symbolic name also “has an associated data format class type corresponding to a subclass of User Interface (UI) objects” (*id.*), as claimed—that is, claimed “User Interface (UI) objects” correspond to the “Text Field” and “Grid Panel” as shown in Figure 10-1 of Anderson (*id.*), reproduced below:



*Anderson's Figure 10-1 depicts Creator's design canvas view.
Ex. 1003, 287.*

Petitioner notes that Figure 10-1, above, shows several user interface (“UI”) objects, including a “Text Field” for receiving a query for a search, and a “Grid Panel” containing “three text output components for displaying results of the search.” Pet. 34–35 (citing Ex. 1003, 286). These UI objects “provide the input and output for the doGoogleSearch method.” *Id.* at 35.

As for the claimed “data format class type,” Petitioner notes that “at least two can be identified based on standard Java classes,” as described in

Geary. *Id.* at 36. In particular, Petitioner contends that Geary teaches standard Java classes for building user interfaces for web applications, including output classes, which belong to the “UIOutput class,” and input components, that belong to the “UIInput class.” *Id.* (citing Ex. 1011, 29, 87). Petitioner then asserts that

[a]s applied to the exemplary web application in Anderson, therefore, it would have been obvious that when building the exemplary web application that accesses the GoogleSearch web service, the symbolic name doGoogleSearch (the name of the search method offered by the web service) “has an associated data format class type corresponding to a subclass of User Interface (UI) objects,” for example, the UIInput or UIOutput classes which would handle, respectively, user input in the “Text Field” for the search query, and the static textual output for the “Grid Panel” of textual elements for displaying search results.

Id. at 37–38 (emphasis omitted) (citing Ex. 1002 ¶ 107).

We note that the ’287 Specification does not mention “data format class type” or “data format type” except in the claims themselves. The Specification does, however, refer repeatedly to various aspects of Java programming when describing the various components of the claimed invention and, in particular, specifically references Java when discussing the claimed classes. *E.g.*, Ex. 1001, 29:50–55 (“If a particular JAVA method is called, which requires an instance of a certain JAVA class to be executed by the run time engine, then that JAVA class is flagged, as well as any supporting methods, variables and/or object definitions.”). Petitioner relies on Anderson, Bowers, and Geary—all of which describe building web applications using Java—as describing User Interface objects that support the data format type of the symbolic name. Pet. 35–40. In particular, Petitioner contends that the example web component with the symbolic

name “doGoogleSearch” has an associated data format class type corresponding to a subclass of User Interface objects—namely, “Text Field” and “Grid Panel” are each user interface objects associated with doGoogleSearch. *Id.* at 34–35. In turn, “Text Field” and “Grid Panel” each correspond to a UIInput or a UIOutput class, which, respectively, handle user input for the “Text Field” search query and the static textual output for the “Grid Panel” elements of doGoogleSearch. *Id.* at 35–39.

Petitioner’s arguments and evidence as to this limitation are persuasive. Patent Owner does not particularly challenge Petitioner’s showing as to this limitation.

Based on Petitioner’s persuasive argument and evidence, we find that the combination of Anderson, Bowers, and Geary teaches or suggests “each symbolic name has an associated data format class type corresponding to a subclass of User Interface (UI) objects that support the data format type of the symbolic name, and has a preferred UI object,” as recited in claim 1.

(c) “an address of the web service”

Claim 1 also recites that the registry stored in computer memory includes “an address of the web service.” Ex. 1001, 37:63.

Petitioner contends Anderson teaches this limitation by disclosing that the GoogleSearch.wsdl file (which Petitioner maps to the claimed “registry,” as discussed *supra* in Section II.F.1.a.(2)) “contains a <service> element pointing to a URL,” and that this URL provides “the address of the web service,” as claimed, “because it provides the ‘address of the actual web service and tells us where to send the request.’” Pet. 42 (citing Ex. 1004, 102). Petitioner notes that, “[i]n this example, the address is

<<http://api.google.com/search/beta2>>.” *Id.* (citing Ex. 1004, 101; Ex. 1002 ¶¶ 98, 115).

Petitioner’s arguments and evidence as to this limitation are persuasive. Patent Owner does not particularly challenge Petitioner’s showing as to this limitation.

Based on Petitioner’s persuasive argument and evidence, we find that Anderson teaches that the registry stored in computer memory includes “an address of the web service,” as recited in claim 1.

(3) “*an authoring tool configured to*”

Claim 1 also recites “an authoring tool configured” to perform several recited functions. Ex. 1001, 37:63.

Petitioner contends that Anderson teaches “an authoring tool” by disclosing “a web application development system known as Creator.” Pet. 42 (citing Ex. 1003, 35). Petitioner contends that “Anderson discloses using the Creator to create a new project for a web application, add graphics and text to the application, and add user interface controls such as a Text Field, Grid Panel, and button for initiating a Google search.” *Id.* at 42–43 (citing Ex. 1003, 286–287). According to Petitioner, “Anderson further describes how to incorporate Google web services by specifying the WSDL file, and incorporating its functionality (including displaying search results) into the web application.” *Id.* at 43 (citing Ex. 1003, 289–296; Ex. 1002 ¶ 116). Petitioner asserts that “Creator qualifies as ‘an authoring tool’ because it provides a user interface that enables a developer to create, generate, and/or modify content and/or functionality within a web application.” *Id.* at 43 (citing Ex. 1002 ¶ 117; Ex. 1009, 6).

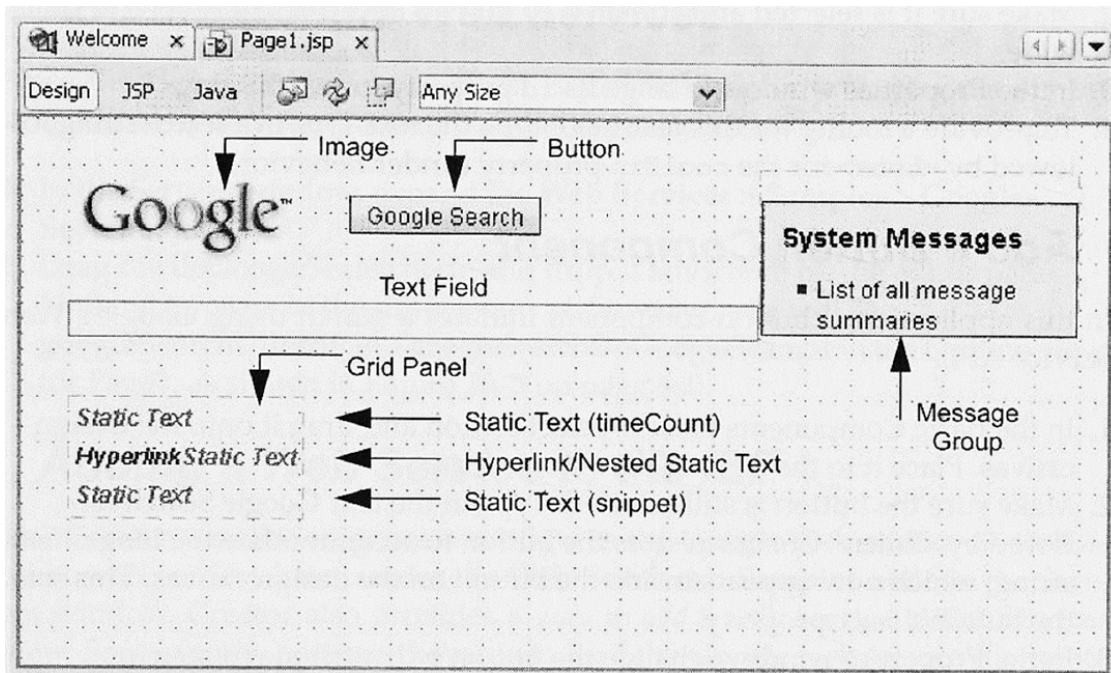
Petitioner's arguments and evidence as to this limitation are persuasive. Patent Owner does not, at this stage, particularly challenge Petitioner's showing as to this limitation. Patent Owner's arguments regarding whether the claims require the authoring tool to produce a player are discussed *supra* § II.C.

Based on Petitioner's persuasive arguments and evidence, we find that Anderson teaches "an authoring tool," as recited in claim 1. Petitioner's showing as to the additional limitations reciting the functionalities of the authoring tool are discussed below.

(a) *"define a (UI) object for presentation on the display . . . "*

Claim 1 also recites that the authoring tool is configured to "define a (UI) object for presentation on the display, where said defined UI object corresponds to a web component included in said registry selected from a group consisting of an input of the web service and an output of the web service." Ex. 1001, 37:64–38:1.

Petitioner contends that Anderson teaches this limitation in connection with its Figure 10-1, reproduced below:



*Anderson's Figure 10-1 depicts Creator's design canvas view.
Ex. 1003, 287.*

Petitioner notes that “Figure 10-1 shows several user interface (UI) objects, including a ‘Text Field’ for receiving the query for a search, and a ‘Grid Panel’ containing three static text output components for displaying results of the search.” Pet. 44 (emphasis omitted) (citing Ex. 1003, 286; Ex. 1002 ¶ 118). Petitioner asserts that each “Text Field” or “Grid Panel” would be a “user interface (UI) object.” *Id.* (emphasis omitted) (citing Ex. 1002 ¶¶ 118–119).

Petitioner argues that the UI objects “correspond 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,” as claimed, because “both the Text Field and Grid Panel correspond to the doGoogleSearch method in the GoogleSearch.wsdl file.” Pet. 45. For confirmation, Petitioner points to Anderson's Figure 10-9, which shows the

“user interface (UI) objects in action executing in a web application that obtains input in the form of a search query, and displays output in the form of search results.” *Id.* at 46.

Petitioner’s arguments and evidence as to defining a UI object are persuasive. Patent Owner does not particularly challenge Petitioner’s showing as to this limitation.

Based on Petitioner’s persuasive argument and evidence, we find that Anderson teaches the authoring tool being configured to “define a (UI) object for presentation on the display, where said defined UI object corresponds to a web component included in said registry selected from a group consisting of an input of the web service and an output of the web service,” as recited in claim 1.

*(b) “where each defined UI object is . . .
selected by a user of the authoring tool”*

Claim 1 also recites

where each defined UI object is either: 1) selected by a user of the authoring tool; or 2) automatically selected by the system as the preferred UI object corresponding to the symbolic name of the web component selected by the user of the authoring tool[.]

Ex. 1001, 38:1–6.

Petitioner contends this limitation is taught by Anderson because it describes how “[t]he Text Field is selected by the user of the authoring tool when the user navigates to the Basic Components palette in Creator,” the user “select[s] component Text field and drag[s] it onto the design canvas,” and the user “[p]lace[s] it below the Google logo.” Pet. 48 (emphasis omitted) (citing Ex. 1003, 287). Petitioner similarly argues that, “[w]ith

respect to the Grid Panel,” the user “select[s] Grid Panel and place[s] it on the page below the text field component.” *Id.* (citing Ex. 1003, 288).

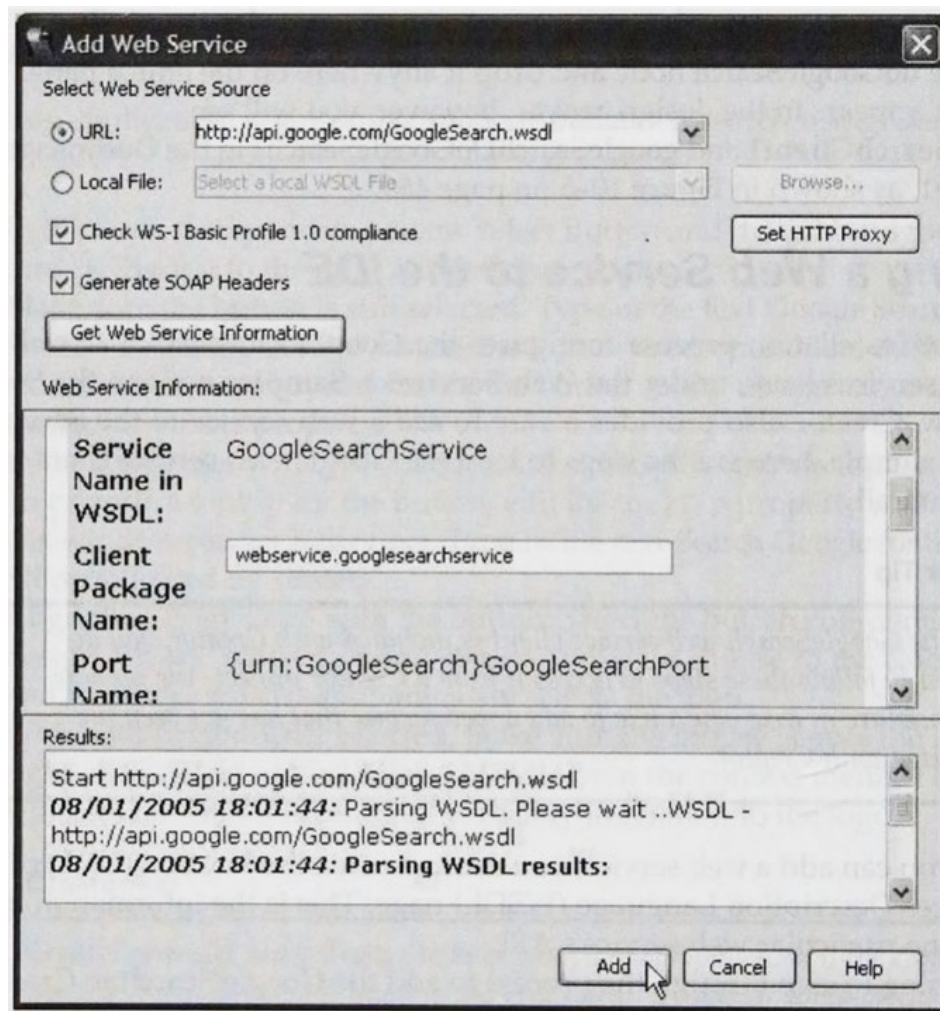
Petitioner’s arguments and evidence as to selecting a defined UI object are persuasive. Patent Owner does not particularly challenge Petitioner’s showing as to this limitation.

Based on Petitioner’s persuasive argument and evidence, we find that Anderson teaches that “each defined UI object is . . . selected by a user of the authoring tool,” as recited in claim 1.

(c) “access said computer memory to select the symbolic name . . .”

Claim 1 also recites that the authoring tool is configured to “access said computer memory to select the symbolic name corresponding to the web component of the defined UI object.” Ex. 1001, 38:7–9.

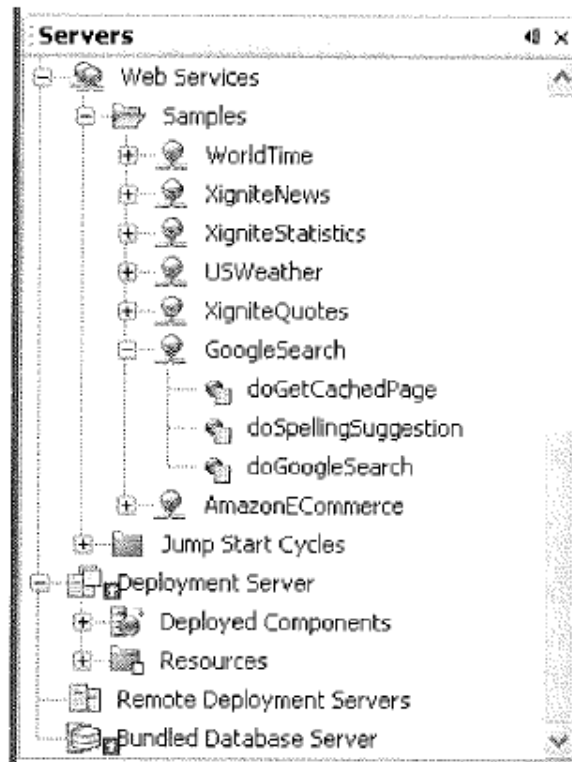
Petitioner argues that “[t]he step of ‘access[ing] said computer memory’ occurs in Anderson when the developer using the authoring tool adds the web service to the current project by (for example) specifying the URL for the GoogleSearch.wsdl file using the Add Web Service dialog in Creator.” Pet. 49–50 (emphasis omitted). Petitioner contends that this is depicted in Anderson’s Figure 10-2, reproduced below.



*Anderson’s Figure 10-2 depicts
Creator’s Add Web Service dialog. Ex. 1003, 290.*

Petitioner argues that “[i]n this example, under ‘Select Web Service Source,’ the developer inputs the URL for Google’s WSDL file,” which “is the GoogleSearch.wsdl file identified for the ‘registry’ limitation of claim 1[a].” Pet. 50–51 (emphasis omitted) (citing Ex. 1002 ¶¶ 77–78, 127). Petitioner argues that Figure 10-2 “further confirms that the authoring tool ‘access[es] said computer memory,’ because it reads the WSDL file stored in the memory and parses it to obtain and display information.” *Id.* at 51 (citing Ex. 1002 ¶ 128). Petitioner further argues that “[t]he access of the

GoogleSearch.wsdl file . . . occurs in Anderson in order ‘to select the symbolic name corresponding to the web component of the defined UI object,’ as claimed,” because the “Add” button “makes the web service available to Creator through another window called Servers.” *Id.* (emphasis omitted) (citing Ex. 1003, 80 & Fig. 2-27; Ex. 1002 ¶¶ 129–130). The Servers window is shown in Figure 2-27, reproduced below:



Anderson’s Figure 2-27 depicts Creator’s Services window. Ex. 1003, 80.

Petitioner’s arguments and evidence as to this limitation are persuasive. Patent Owner does not particularly challenge Petitioner’s showing as to this limitation.

Based on Petitioner’s persuasive argument and evidence, we find that Anderson teaches “access[ing] . . . computer memory to select the symbolic

name corresponding to the web component of the defined UI object,” as recited in claim 1.¹⁸

(d) “associate the selected symbolic name with the defined UI object . . .”

Claim 1 also recites that the authoring tool is configured to “associate the selected symbolic name with the defined UI object, where the selected symbolic name is only available to UI objects that support the defined data format associated with that symbolic name.” Ex. 1001, 38:10–14.

Petitioner argues that this limitation is taught in Anderson because the Grid Panel “is associated with doGoogleSearch because it displays the output of that method” and the Text Field “is also associated with doGoogleSearch because the text input by the user will provide parameter ‘q,’ i.e., the textual search query for doGoogleSearch.” Pet. 53–54 (emphasis omitted) (citing Ex. 1003, 287–88, 303; Ex. 1002 ¶ 132). Petitioner further argues that “[t]he symbolic name is ‘available’ to those UI objects because . . . the software links these objects to the doGoogleSearch symbolic name by assigning them roles in the input and output of the doGoogleSearch method as explained above.” *Id.* at 54 (emphasis omitted). Petitioner also argues that “a skilled artisan would have found this limitation to state nothing more than a basic principle of software design in an object-

¹⁸ We also find that the user is “accessing computer memory” to select the symbolic name when using Creator’s Server dialog window when the WSDL file is local. The information that is displayed in the dialog box (e.g., the names of the services) must be stored in the computer memory in order to be listed in the dialog box, and the dialog box would allow the user to “access” that stored information (e.g., to select a service). Additionally, it appears that, when viewing the image displayed on the monitor or screen showing the dialog box, the user would be “accessing” the computer’s video memory, which is also “computer memory,” in order to select a name.

oriented system, *i.e.*, that the only information that should be made available to an object is the specific information it needs to carry out its purposes,” and that it would have been obvious for the doGoogleSearch symbolic name to have been available only to UI objects that “play a role in submitting input to, or presenting output from, the doGoogleSearch method . . . as there would be no reason for the name to be available to other UI objects that lack any such role.” *Id.* at 57–58 (citing Ex. 1002 ¶ 35).

Petitioner’s arguments and evidence as to associating the symbolic name with the defined UI object are persuasive. Patent Owner does not particularly challenge Petitioner’s showing as to this limitation.

Based on Petitioner’s persuasive argument and evidence, we find that Anderson teaches “associat[ing] the selected symbolic name with the defined UI object, where the selected symbolic name is only available to UI objects that support the defined data format associated with that symbolic name,” as recited in claim 1.

(e) “produce an application including the selected symbolic name of the defined UI object, where said Application is a device-independent code”

Claim 1 also recites that the authoring tool is configured to “produce an Application including the selected symbolic name of the defined UI object, where said Application is a device-independent code.” Ex. 1001, 38:15–17.

Petitioner argues that this is described in Anderson, which “explains that upon selecting the ‘Run’ option from the user interface, ‘Creator builds the application, deploys it, and brings up a browser with the [developer’s] web application running.’” Pet. 58 (emphasis omitted) (quoting Ex. 1003, 87). Petitioner argues that the application includes the selected symbolic

name because “the whole purpose of the application is to invoke doGoogleSearch (‘the selected symbolic name’) to perform a search.” *Id.* (emphasis omitted) (citing Ex. 1002 ¶¶ 136–137). Petitioner also argues it “would have been obvious that the application would include the selected symbolic name because the application must include the doGoogleSearch symbolic name to construct and issue a proper message to the GoogleSearch web service requesting performance of the method.” *Id.* at 59 (emphasis, internal quotation marks, and brackets omitted) (citing Ex. 1002 ¶ 137).

Petitioner further argues that the Java application “qualifies as ‘a device-independent code,’ because bytecode instructions can run on any processor that has a Java Virtual Machine (JVM).” Pet. 60 (emphasis omitted) (citing Ex. 1002 ¶ 139).

Patent Owner argues that Petitioner’s proposed combination is deficient because the combination fails to teach any application that is device-independent code. PO Resp. 12. Specifically, Patent Owner asserts that because “Petitioner and its expert admit that their proposed Application is ‘written in the Java programming language,’” compiled into Java bytecode, and “run on a Java virtual machine,” “the alleged application is specific to the Java platform and is not device-independent code as recognized in the specification and under agreed and court-ordered construction of the term.” PO Resp. 13–14.

According to Patent Owner, “[t]he Java website further demonstrates that Petitioner’s proposed application is specific to a platform” because it “recognizes that ‘Java capability for mobile devices is generally integrated by the device manufacturers [and] is NOT available for download or installation by consumers.’” See PO Resp. 14 (citing Ex. 2028, 1). Patent

Owner contends that, because “Petitioner’s proposed Java application is only compatible with the Java platform that must be included by a manufacturer on a device,” it “is a far cry from the device-independent solution claimed by the ’287 patent.” *Id.* at 15.

Patent Owner contends that Petitioner’s argument that “Anderson’s application is device-independent because it can allegedly ‘run on any processor that has a Java Virtual Machine (JVM)’ . . . misses the point” because, “[a]s taught in the specification and demonstrated by the agreed construction of the parties, code is only device-independent when it is not specific to a particular ‘operating system, programming language, or platform’ of a device.” PO Resp. 15 (citing Pet. 60; Ex. 2021, 15–16; Ex. 1001, 3:59–62, 6:55–57).

Finally, Patent Owner argues that “Petitioner’s citation to the specification for the proposition that an Application may include Java programming . . . misses the mark” because “[t]he claim requires not only an Application, but an Application that is device-independent code.” PO Resp. 15. According to Patent Owner, “[w]hile an Application may include Java programming, the specification [teaches] an embodiment of converting a device-dependent Java application to an Application that is device-independent code by leveraging PDL.” *Id.* (citing Ex. 1001, 6:8–7:10; Ex. 2020 ¶ 55).¹⁹ Patent Owner contends that “[t]he specification teaches that an Application with internal programming of Java can become device-independent by having its ‘Java code translated into a PDL,’” which “can

¹⁹ PDL, as used in the ’287 patent, refers to “Portable Description Language.” Ex. 1001, 6:10.

make the Application device-independent because ‘the reassembled primitives of the Application no longer have any dependencies from the original programming language (Java) that they were defined in.’” *Id.* (citing Ex. 1001, 7:21–25).

Petitioner responds that “Patent Owner’s arguments cannot be reconciled with the specification of its own patent, which describes ‘**device-independent code**’ and then expressly identifies a Java application as an example of such code.” Reply 2 (citing Pet. 60–61; Ex. 1002 ¶ 139).

Regarding the PDL, Petitioner argues that “the specification does not describe a Java program as merely a generic application, but as an example of an ‘Application’ – expressly referencing the patent’s own specialized description of an ‘Application’ (with capital ‘A’) as a device-independent program.” Reply 4 (citing Ex. 1001, 5:60–6:1, 5:13–19).

Petitioner contends that “[t]he fact that a Java program must be interpreted by ‘the Player’ further confirms that the program is ‘**device-independent code**’” because “the Player transforms device-independent instructions of the Application into device-specific instructions that are executable by device 130,” and “[i]f Java programming was device-dependent code . . . it would not need the Player to transform its ‘device-independent instructions . . . into device specific instructions.’” Reply 4 (emphasis omitted) (citing Ex. 1001, 5:60–6:1; Ex. 1021 ¶ 16). Petitioner further argues that “Patent Owner’s reliance on PDL also fails because PDL merely refers to an exemplary embodiment that does not limit the scope of the claimed ‘**Application.**’” *Id.* at 5. Petitioner additionally argues that “[t]he prior art also confirms that Java programs are made up of bytecode instructions that are device-independent” and that “Patent Owner’s

interpretation of the proposed construction would also render the claims inoperable because no program could ever have the required ‘device-independent code.’” *See id.* at 5–12.

We agree with Petitioner that the combination discloses “code that is not specific to the operating system, programming language, or platform of a device,” namely, the Java application produced by Anderson’s authoring tool. *See* Pet. 58–61. The record shows that Java is device-independent code, and it appears that device independence is, in fact, one of the primary benefits of programming in Java. *See* Ex. 1022, 5 (“Java was . . . designed to be secure and platform neutral (meaning that it can be run on any platform) through the fact that Java programs are compiled into bytecode which is not refined to the point of relying on platform specific instructions.”); Ex. 1023 (“Java is a semi-compiled language that is compiled down to a platform-independent bytecode which can be run on any kind of processor.”); Ex. 1003, 35 (“Java includes a compiler that produces portable bytecode and a Java Virtual Machine (JVM) that runs this bytecode on any processor.”); Ex. 1002 ¶ 139.

We are unpersuaded by Patent Owner’s argument that Java is not “device independent” because the agreed-to construction of “device independent” requires that it be “code that is not specific to [a] platform of a device.” This is because we find that Java is simply a programming environment that allows a single application to work across many devices, and it not, itself, a “platform of a device.” Although Petitioner characterizes Java as a “platform,” the evidence it cites refers to Java as a “*technology* platform,” not a platform *of a device*. *See* Ex. 1003, 35. And Petitioner’s expert opines only that “Anderson teaches that its proposed Java application

is specific to the Java programming language and the particular disclosed *Java* platform,” not a platform of a device. Ex. 2020 ¶ 50.

Moreover, even if a particular JVM could be characterized as a “platform of a device,” Anderson’s Java applications would *not* be specific to *that* platform, because, as noted above, one of the main benefits of Java is that it will run on JVMs on *any* device. Java applications are specific to Java, but not to any device or any particular JVM.

We thus conclude that Anderson’s Java applications *are* “code that is not specific to the operating system, programming language, or platform of a device.” This is fully consistent with the ’287 patent’s explanation that its “Application” with device-independent instructions “may include Java programming.” *See* Ex. 1001, 5:60–6:1.

In fact, we conclude that this is the only reasonable resolution of this issue because Patent Owner’s position on the application of the “device independent” construction would essentially mean that *no* applications could be device-independent. Code written to run directly on the hardware of a device would not be “device-independent,” because it would have been written for that specific hardware, but code running at any higher level of abstraction, and thus requiring interpretation, would, under Patent Owner’s theory, be “specific to [a] platform of a device,” *i.e.*, the interpreter, and thus also not “device-independent.” That simply cannot be correct.

We are also unpersuaded by Patent Owner’s argument about the PDL because we agree with Petitioner that use of the PDL is described in the Specification as optional, and there is nothing to suggest that the claims are limited to that embodiment. *See* Ex. 1001, 6:54–55 (noting that, “[i]n

certain embodiments, the applicant model in developing and providing Applications is a PDL”).

Based on Petitioner’s persuasive argument and evidence, we find that Anderson teaches “produc[ing] an Application including the selected symbolic name of the defined UI object, where said Application is a device-independent code,” as recited in claim 1.

(4) “*a Player, where said Player is device-dependent code . . .*”

Claim 1 also recites “a Player, where said Player is device-dependent code.” Ex. 1001, 38:19–20.

Petitioner argues that this limitation reads on a Java Virtual Machine (“JVM”), as taught by Anderson and Ambrose-Haynes. Pet. 61–63. In particular, Petitioner contends that “the authoring tool under the proposed combination produces a Java application (in the form of device-independent bytecode) which is interpreted and executed by the JVM.” Pet. 61.

Petitioner asserts that this is confirmed by Ambrose-Haynes, which Petitioner states “provides additional detail about the JVM mentioned in Anderson.” *Id.* Ambrose-Haynes describes JVMs as linked to different operating systems:

Code compiled from Java is not executable code; rather, it’s byte code. Byte code is a set of instructions that is interpreted within a Java Runtime Environment (JRE), consisting of the core Java class and support files, as well as a Java Virtual Machine (JVM), which actually executes the byte code. Executing the code through a JVM solves the problem of portability, since *JVMs are written for each operating system*. In this way *it is the JVM which is linked to the operating system*, and handles all of the platform-specific details, while the source code you write for interpretation is platform independent.

Ex. 1006, 55 (emphases added). Petitioner asserts that the JVM, as

described in both Anderson and Ambrose-Haynes, is “device-dependent code,” because it is software directly executing on the computer and is tailored according to particular operating systems for different computers. *See* Pet. 61–62.

Patent Owner presents two arguments against Petitioner’s showing as to this limitation. PO Resp. 16–26.

Patent Owner first argues that Petitioner’s showing is deficient because a Java virtual machine is not a “Player.” PO Resp. 17. Specifically, Patent Owner argues that “the specification makes clear that a Player is not a virtual machine,” because it describes “a Player and virtual machine as separate components with separate functionalities and even illustrates the two components separately.” *Id.* (citing Ex. 2020 ¶ 57). Patent Owner argues that “Petitioner’s theory, where a Player is a Java virtual machine, ignores [various] disclosed embodiments and is thus contrary to the teachings of the specification.” *Id.* (referring to Ex. 1001, 11:65–12:9, 1:63–67, 11:66–2:9, 6:4–7, 7:21–28).

Patent Owner also argues that one of ordinary skill in the art “would understand that a Java virtual machine does not disclose the claimed Player” because “[a] Java virtual machine receives Java bytecode, which is binary code specific to Java,” but, “as the inventors recognized during prosecution of the parent application, the claimed Player may, when implemented in an embodiment that interprets the Application, ‘interpret non-binary, device independent [code] contained in the Application.’” PO Resp. 18–19 (citing Ex. 2037, 9).

Patent Owner further contends that “[a] Player is not a machine, or a machine emulated in software,” as, “instead, a Player facilitates the

execution of an Application on a machine.” PO Resp. 19. According to Patent Owner, “the entire purpose of the Player is to facilitate the execution of an Application on a particular machine, whether it be physical or virtual.” *Id.*

Petitioner responds that “the Java Virtual Machine (JVM) qualifies as a ‘Player’ because it carries out all functions of the Java application.” Reply 12 (citing Pet. 61–62, 75–77; Ex. 1002 ¶¶ 59–60, 165; Ex. 1021 ¶ 33). According to Petitioner, “[i]n every litigation in which it has asserted the challenged patent, Patent Owner has consistently argued that a Player is simply ‘software code that facilitates the execution of an application on a device.’” *Id.* (citing Pet. 11; Ex. 1024, 4; Ex. 1025, 9; Ex. 1026, 15; Ex. 1027, 13).

Regarding the argument that “[a] Player is not a machine, or a machine emulated in software [because] . . . a Player facilitates the execution of an Application on a machine,” Petitioner asserts that “[the] concepts are not mutually exclusive; something that calls itself a virtual machine can also ‘facilitate[] the execution of an Application on a device, under Patent Owner’s construction, and the JVM does precisely that.” Reply 13 (citing Pet. 61–62, 75–77; Ex. 1021 ¶ 34).

Petitioner further argues that “the intrinsic support does not support” a “new negative limitation on ‘Player’ excluding any software component labeled a ‘virtual machine.’” Reply 13. Specifically, Petitioner contends that Figure 2B of the patent is “simply an embodiment,” and that “[t]he specification also discloses a far more pertinent embodiment in which [the virtual machine] is not required because [the Player] is a native program that communicates directly” with the operating system. *See id.* at 14–15 (citing

Ex. 1001, 12:10–11). Petitioner contends that “[t]he challenged claims reflect this embodiment by reciting a native or device-dependent ‘Player’ and making no mention of any virtual machine.” *Id.* at 15. According to Petitioner, “[t]his alternative embodiment mirrors the functions of the JVM, which is a device-dependent / native program that communicates with the operating system of the device.” *Id.* (citing Ex. 1021 ¶ 32; Ex. 1006, 55; Ex. 1002 ¶¶ 60, 142–144; Pet. 61).

We agree with Petitioner. The Petition identifies two constructions for the term “Player,” one attributed to Petitioner and one attributed to Patent Owner, and alleges that the combination meets either construction. *See* Pet. 10–11 (constructions), 61–62 (arguing the JVM is a Player). We agree that the JVM meets either construction, as it is “device specific code, that is separate from the Application and interprets or executes the Application,” and also is “software code that facilitates the execution of an application on a device.” *See* Pet. 11.

Patent Owner does not argue that the constructions identified by Petitioner do not cover a JVM.²⁰ Instead, Patent Owner asserts that *one of skill in the art would understand* that a JVM is not a “Player” because (a) the specification describes the Player and a virtual machine as separate components with separate functionalities; and (b) a JVM receives Java bytecode but the prosecution history indicates that the Player may “interpret

²⁰ *See* Tr. 43 (counsel confirming Patent Owner’s agreement with the *Shopify* court’s construction of “Player” as “device-specific code which contains instructions for a device and which is separate from the Application”).

non-binary, device independent [code] contained in the Application.”

PO Resp. 17–19.

These arguments are unpersuasive because it is insufficient for Patent Owner to simply point to aspects of embodiments or excerpts from the prosecution history. We are not comparing the specification or the file history to the prior art. Instead, our task is to *construe the claims* in view of the intrinsic evidence and then compare the construed claims to the prior art. *See, e.g., Enzo Biochem, Inc. v. Applera Corp.*, 599 F.3d 1325, 1332 (Fed. Cir. 2010). Here, the only possible constructions that have been identified are in the Petition, and we find that both of those would cover a JVM. Patent Owner’s arguments fail because Patent Owner has not offered a construction of “Player” that would be limited in the ways Patent Owner suggests, much less provided sufficient analysis to have us adopt any such construction.

We further find that the arguments Patent Owner makes would have been insufficient to support a limiting construction in any event. In particular, we agree with Petitioner that Figure 2B of the Specification shows just one embodiment,²¹ and that the Specification describes the “Player” more broadly as something executing “device- or device-platform specific instructions for processor . . . of the device.” Ex. 1001, 5:16–18; *see id.* at 5:60–63 (describing how, “for example and without limitation, in one embodiment, the Player transforms device-independent instructions of the Application into device-specific instructions that are executable by

²¹ *See, e.g.,* Ex. 1001, 2:24–26 (“FIG. 2B is a schematic of one embodiment of a device illustrating an embodiment of the programming generated by authoring platform.”).

device 130”). In fact, the specification describes how “the Application may include Java programming for generating a display on Screen 137,” and that “the Player may *interpret the Java* and instruct processor 135 to produce the display according to the Application,” *id.* at 5:64–67 (emphasis added), strongly suggesting that the Player could be a JVM, or at least software with very similar capabilities.

Patent Owner’s arguments concerning the file history do not fare better. Patent Owner does not provide any support for its assertion that “Java bytecode” is “binary code” other than the word of its expert. *See* PO Resp. 19 (citing Ex. 2020 ¶ 58). We conclude that Java bytecode is *not* “binary code,” at least as that term is used in its most strict sense, because Java “[b]yte code is a set of instructions that is interpreted within a Java Runtime Environment (JRE).” Ex. 1006, 55; *see* Ex. 1002 ¶ 139 (“Java applications instead comprise device-independent instructions (bytecode) that can execute on any processor”). We find that the prosecution argument that “the Player . . . interpret[s] non-binary, device independent contained in the Application,” was simply making the point that the Player did not execute low level, machine specific code, not that it would not execute Java bytecode.

Finally, we are unpersuaded by Patent Owner’s argument that a Player is “not a machine, or a machine emulated in software” because it “facilitates the execution of an Application on a machine.” PO Resp. 19. This argument fails for the simple reason that a virtual machine *does* “facilitate[] the execution of an Application on a machine,” namely the machine on which the JVM is running.

For these reasons, we conclude that Petitioner has shown that a JVM may be a “Player,” as claimed.

Patent Owner next argues that the Petition is deficient with regard to this limitation because it fails to show that the Player is *produced by the authoring tool*, as Petitioner has argued in claim construction arguments in underlying district court litigation. PO Resp. 16, 20–26. We address this argument *supra* in Section II.C. As explained therein, we decline effectively to add words to claim 1 and thus require that the authoring tool must “produce” the claimed Player. Therefore, Patent Owner’s arguments regarding Petitioner’s alleged failure to show that the authoring tool produces the player are inapposite.

Petitioner’s arguments and evidence as to this limitation are persuasive. Patent Owner’s challenges to Petitioner’s showing are based on a misreading of the record and an incorrect claim construction, and hence do not undermine Petitioner’s persuasive showing.

Based on Petitioner’s persuasive argument and evidence, we find that Petitioner has demonstrated that the cited art teaches or suggests “a Player, where said Player is device-dependent code,” as recited in claim 1.

(a) “*wherein . . . when the user of the device provides one or more input values associated with an input symbolic name . . .*”

Claim 1 also recites, in connection with the claimed Player, “wherein, when the Application and Player are provided to the device and executed on the device, and when the user of the device provides one or more input values associated with an input symbolic name to an input of the defined UI object,” several listed events happen. Ex. 1001, 38:20–25.

Petitioner argues that this limitation is taught by the proposed combination of Anderson and Ambrose-Haynes, in which the Java application of Anderson, built under the proposed combination (corresponding to the claimed “Application”), executes on a device using the JVM described in both Anderson and Ambrose-Haynes (corresponding to the claimed “Player”), in order to provide a search feature using doGoogleSearch. Pet. 63–64 (citing Ex. 1003, 76–77 (describing building application using Creator and running application on same machine (or different machine)), 303 (describing using same machine (“localhost”) to run GoogleSearch application)). Petitioner asserts that Figure 10-9 of Anderson, reproduced below, shows a user providing a search query in the text box (corresponding to “one or more input values . . . to an input of the defined UI object”).

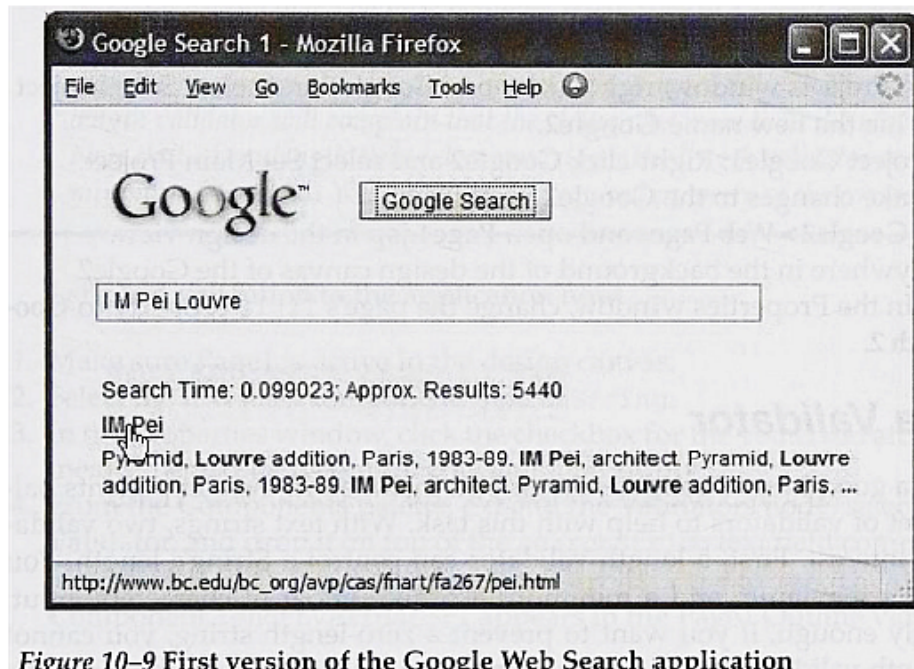


Figure 10–9 First version of the Google Web Search application

Anderson’s Figure 10-9 depicts a Google Web Search application. Ex. 1003, 307.

Referring to Figure 10-9, above, Petitioner asserts that the search query “I M Pei Louvre” corresponds to the claimed “one or more input values,” and is provided to the Text Field (corresponding to the claimed “defined UI object”). Pet. 64. Petitioner further asserts that these input values are “associated with an input symbolic name” because they are associated with doGoogleSearch, the name of the method that receives the query and carries out the search. *Id.* (citing Ex. 1002 ¶¶ 122, 152).

Petitioner’s arguments are persuasive and are supported by the cited evidence. Patent Owner does not particularly challenge Petitioner’s showing as to this limitation.

Based on Petitioner’s persuasive argument and evidence, we find that the proposed combination teaches or suggests “wherein . . . when the user of the device provides one or more input values associated with an input symbolic name,” as recited in claim 1.

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

Claim 1 also recites, in connection with the claimed Player, that one of the listed events is “1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service.” Ex. 1001, 38:26–28.

Petitioner argues that this limitation is taught by the disclosure in Anderson, and in particular Anderson’s description of the method doGoogleSearch, which teaches that when the user of the Java application enters a search query into the text box and presses “Google Search” (as depicted in Figure 10-9 of Anderson), the search query is provided to the doGoogleSearch method of the web service. Pet. 65. Relying on code

reproduced below from Bowers (which also describes the same doGoogleSearch method as described in Anderson), Petitioner contends that the GoogleSearch.wsdl file specifies that the search query is provided via an “input message” for the doGoogleSearch method.

```
<operation name="doGoogleSearch">  
  <input message="typens:doGoogleSearch"/>  
  <output message="typens:doGoogleSearchResponse"/>  
</operation>
```

*The image above, as annotated by Petitioner, is a portion of a WSDL document describing Google Web APIs.
See Pet. 65; Ex. 1004, 102.*

As depicted above, the “doGoogleSearch” method allows a search query to be input via an “input message” (also named “doGoogleSearch”), annotated above with a red box, and provides a corresponding “output message” (named “doGoogleSearchResponse”). As Petitioner notes, the “doGoogleSearch” message type in the WSDL file, in turn, “defines a number of input parameters, including ‘q’ which . . . contains the desired search query string.” Pet. 65–66 (citing Ex. 1004, 102). As Bowers explains, “[t]his definition [for doGoogleSearch] tells us the message name and the names and data types of each parameter So, the WSDL file, among other things, defines the format and input types of all the messages the service listens to.” Ex. 1004, 102. Based on this disclosure, Petitioner asserts that “[w]hen invoking the 'doGoogleSearch method, therefore, the device provides an input message to the web service, the message identifying (among other things) the search query ‘q’ (‘the user provided one or more input values’) and the name of the ‘doGoogleSearch’ message (‘and

corresponding input symbolic name’).” Pet. 66 (emphasis omitted) (citing Ex. 1002 ¶ 154).

Petitioner additionally points to Jacobs as confirming the foregoing by providing “a concrete example of a doGoogleSearch input message communicated to the Google web service in order to run an actual search.” *Id.* at 66–67 (citing Ex. 1005, 104; Ex. 1002 ¶¶ 154–155).

Petitioner’s arguments as to this limitation are persuasive and are supported by the cited evidence. Patent Owner does not particularly challenge Petitioner’s showing as to this limitation.

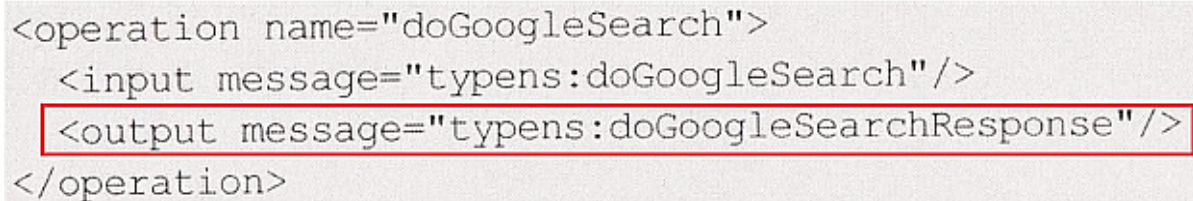
Based on Petitioner’s persuasive argument and evidence, we find that Anderson, in light of Bowers and Jacobs, teaches or suggests “1) the device provides the user provided one or more input values and corresponding input symbolic name to the web service,” as recited in claim 1.

(c) “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 . . .”

Claim 1 also recites, in connection with the claimed Player, that one of the listed events is “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.” Ex. 1001, 38:29–32.

Petitioner argues that this limitation is taught by the disclosure in Anderson and Bowers regarding the GoogleSearch web service that utilizes the doGoogleSearch input message (which Petitioner contends contains both the input symbolic name and the user provided input values) to generate an output message that includes output search results based on the search query (which Petitioner contends corresponds to the claimed “one or more output

values”). Pet. 70–75. In particular, Petitioner contends that “the GoogleSearch.wsdl file specifies an output message type used to communicate . . . search results to the application, *i.e.*, ‘doGoogleSearchResponse.’” *Id.* at 70. Petitioner provides an excerpt from Bowers, reproduced below with annotation provided by Petitioner, which illustrates this point:



```
<operation name="doGoogleSearch">
  <input message="typens:doGoogleSearch"/>
  <output message="typens:doGoogleSearchResponse"/>
</operation>
```

*The image above, as annotated by Petitioner, is a portion of a WSDL document describing Google Web APIs.
See Pet. 70; Ex. 1004, 102.*

Petitioner notes that the “output message” depicted above, in the red box, is identified as “doGoogleSearchResponse,” which Petitioner contends corresponds to the claimed “output symbolic name,” and is associated with the output search results from the doGoogleSearch method. Pet. 70–71 (citing Ex. 1003, 297; Ex. 1004, 100). Petitioner further notes that, as described in Anderson, the search returns a “GoogleSearchResult object,” which is associated with “a number of output values.” *Id.* at 71–72 (citing Ex. 1003, 297, 299, Table 10.2). Petitioner then summarizes:

Anderson thus confirms that the web service utilizes doGoogleSearch and the search query (“the input symbolic name and the user provided one or more input values”) to generate a doGoogleSearchResponse output message containing search results (“for generating one or more output values”).) Because the output message is specifically named in the GoogleSearch.wsdl file by its symbolic name (doGoogleSearchResponse), the “one or more output values

hav[e] an associated output symbolic name,” as recited in the claim.

Pet. 73 (emphasis omitted) (citing Ex. 1002 ¶¶ 160–161).

Petitioner additionally points to Jacobs as confirming the foregoing by providing a concrete example of a doGoogleSearch output message communicated from the Google web service to the application using SOAP. *Id.* at 73–74 (citing Ex. 1005, 105; Ex. 1003, 299; Ex. 1002 ¶¶ 156–158, 162).

Petitioner’s arguments and evidence as to this limitation are persuasive. Patent Owner does not particularly challenge Petitioner’s showing as to this limitation.

Based on Petitioner’s persuasive argument and evidence, we find that Anderson teaches 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 recited in claim 1.

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

Claim 1 finally recites, also in connection with the claimed Player, that one of the listed events is “(3) said Player receives the output symbolic name and corresponding one or more output values and provides instructions for the display of the device to present an output value in the defined UI object.” Ex. 1001, 38:33–36.

Petitioner argues that this limitation is taught by the combined disclosures of Anderson and Bowers with regard to the “doGoogleSearch” method described above. Pet. 75–78. In particular, Petitioner contends that

after the search has been performed, the GoogleSearch web service sends the doGoogleSearchResponse output message to the Java application. *Id.* at 75. Petitioner notes that the “output message . . . identifies itself using its symbolic name and includes the search results, thus confirming that the Java application ‘receives the output symbolic name and corresponding one or more output values.’” *Id.* (emphasis omitted). Petitioner further reasons that because it is the JVM (corresponding to the claimed “said Player”) that executes the Java application on the device, the output message object is received by the JVM in order to be made available to the Java application. *Id.* (citing Ex. 1006, 55 (“[T]he JVM on any given platform will interpret and execute the [Java application’s] code for you.”)). Thus, according to Petitioner, “[t]he Player (as well as the Application) . . . ‘receives the output symbolic name and corresponding one or more output values,’ as claimed.” *Id.* (emphasis omitted) (citing Ex. 1002 ¶ 163).

Petitioner further alleges that the Player described in Anderson and Bowers also “provide[s] instructions for the display of the device to present an output value in the defined UI object,” because “it carries out the functions of the Java application relating to displaying the user interface.” *Id.* at 75–76 (emphasis omitted). Petitioner refers to Figure 10-1 of Anderson, reproduced below as annotated by Petitioner:

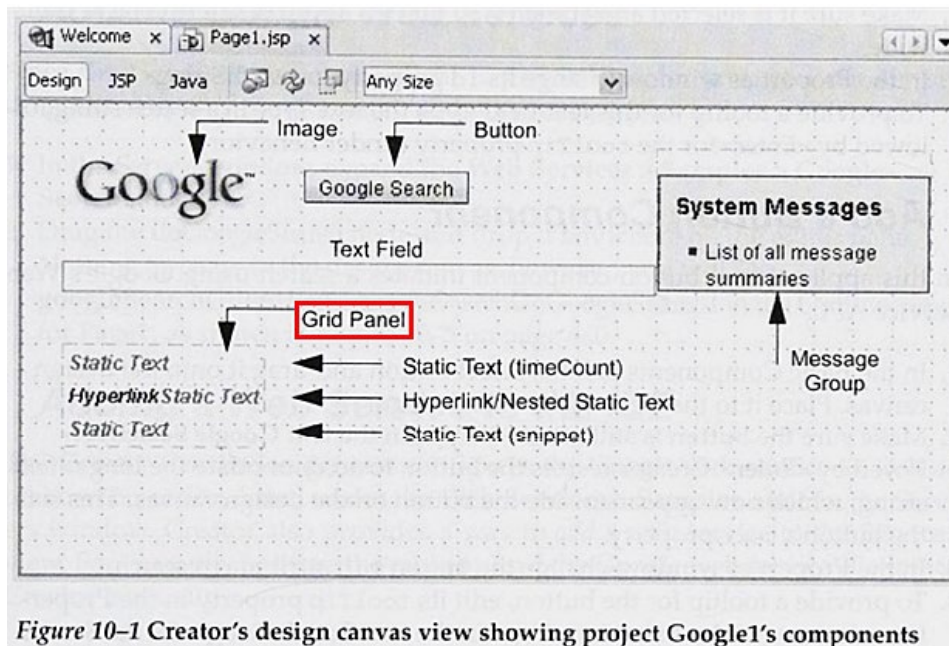


Figure 10-1 Creator's design canvas view showing project Google1's components

Anderson's Figure 10-1, as annotated by Petitioner, depicts Creator's design canvas view. Pet. 76; Ex. 1003, 287.

Petitioner asserts that the Grid Panel shown in Figure 10-1 (corresponding to the claimed "defined UI object") presents the search results obtained from the doGoogleSearch method (corresponding to the claimed "present an output value in the defined UI object"). Pet. 76 (citing Ex. 1003, 303 ("The static text and hyperlink components that you nested inside the grid panel display portions of the result returned from the call to Google's search method.")) (emphasis omitted). Figure 10-9 of Anderson, reproduced below, shows the running Java application with the "Grid Panel" displaying these results.

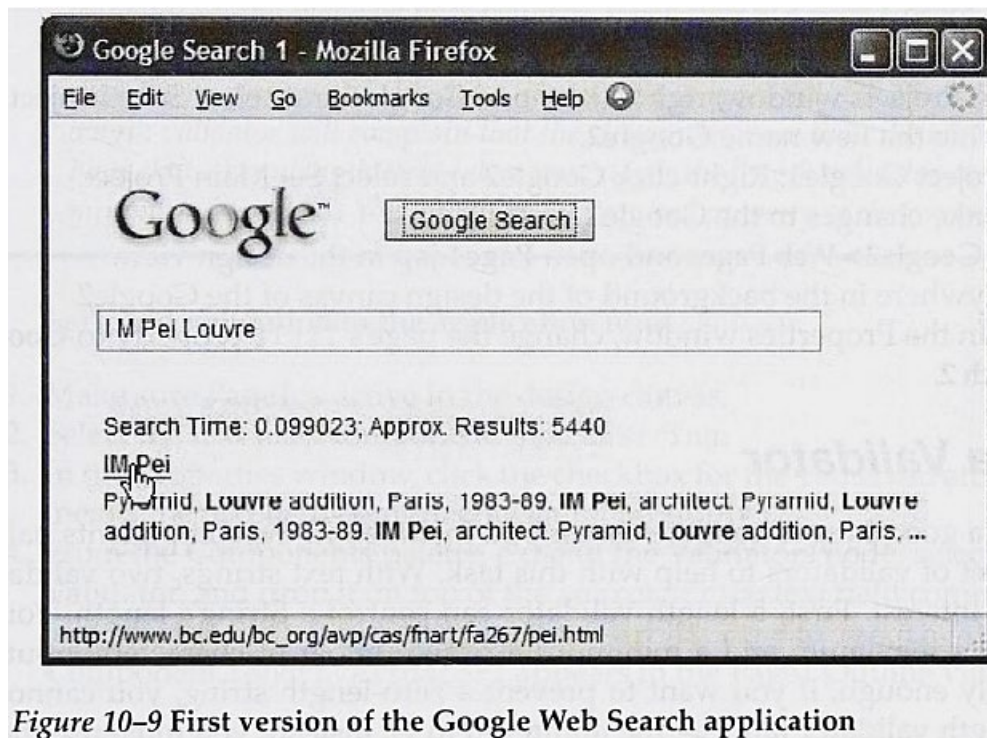


Figure 10-9 First version of the Google Web Search application

*Anderson's Figure 10-9 depicts Creator's design canvas view.
Ex. 1003, 307.*

According to Petitioner:

Under the proposed combination, the Java Virtual Machine (JVM) ("said Player") is responsible for carrying out all functions of the Java application. . . . It therefore would have been obvious that because the Java application requires the JVM to execute, the JVM ("said Player"), as well as the application, receives the output message from the web service ("output symbolic name and corresponding one or more output values") and provides instructions to display the search results on the display of the device ("provides instructions for the display of the device to present an output value in the defined UI object").

Pet. 77-78 (emphasis omitted) (citing Ex. 1006, 55; Ex. 1002 ¶ 165).

Patent Owner argues that the proposed combination fails to teach this limitation because "Anderson and Ambrose-Haynes teach that a Web server provides server-side processing and returns an HTML page to the browser of

the client-side device,” and thus “in the proposed combination, the Web browser, and not the alleged Player, would provide any instructions for the display of the device to present any output values or other display.”

PO Resp. 32–33 (citing Ex. 1003, 92; Ex. 1006, 45; Ex. 2020 (Second Almeroth Decl.) ¶ 73).

Petitioner responds by noting that nothing in the challenged claims requires that the Player itself directly present or physically draw the user interface for the user; the claim requires only that the Player provide “instructions” for such display. Reply 21–22. Petitioner then explains that, in Anderson, the JVM (“Player”) on the device carries out the functions of the Java application, resulting in instructions that cause display of the user interface on the web browser. Reply 21–22 (citing Ex. 1002 ¶¶ 164–165; Ex. 1003, Figs. 10-1, 10-9 (depicting output in web browser generated from instructions received by the JVM running on the developer computer); *see also* Pet. 75–78. Petitioner’s arguments are persuasive and supported by the cited evidence.

Based on Petitioner’s persuasive argument and evidence, we find that the cited prior art teaches that “said Player receives the output symbolic name and corresponding one or more output values and provides instructions for the display of the device to present an output value in the defined UI object,” as recited in claim 1.

(5) Motivation to Combine

Petitioner offers a series of reasons that it contends would have motivated the ordinarily skilled artisan to combine Anderson, Bowers, Jacobs, Ambrose-Haynes, and Geary. *See* Pet. 20–21 (Anderson and Bowers), 40–42 (Anderson and Bowers), 62–63 (Anderson and Ambrose-

Haynes). Essentially, Petitioner uses Anderson as a base reference and asserts that it would have been obvious to add Bowers “for the contents of the Google WSDL file identified in Anderson to confirm what a skilled artisan would have already appreciated – that the GoogleSearch.wsdl file in fact, is an XML file and thus qualifies as a registry.” Pet. 20. Petitioner then argues it would have been obvious to add Geary, which shows that a “Java web application built in accordance with the teachings of Anderson would have included classes/subclasses for user interface objects that receive textual input and handle textual output for the doGoogleSearch method.” *Id.* at 40. Finally, Petitioner adds “Ambrose-Haynes largely to confirm what a skilled artisan would have understood about Java, *i.e.*, that the JVM is device-dependent code.” *Id.* at 62. The secondary references are thus offered more to provide details that are omitted from Anderson than as the basis for any modification of Anderson. Petitioner also points to Jacobs as providing concrete examples of using the Google web service in a manner that Petitioner contends confirms Petitioner’s contentions. *E.g., id.* at 66–69, 73–75.

Patent Owner presents several arguments against the proposed combination, which we address in turn below.

(a) Patent Owner’s argument that it would not have been obvious to implement the proposed combination in a client-side framework

First, Patent Owner argues that it would not have been obvious to implement the proposed combination in the client-side framework required under Petitioner’s proposed combination, in which the authoring tool, Application, and Player all run on the same computer. *See* PO Resp. 26–34. In particular, Patent Owner contends that Anderson and Ambrose-Haynes

both disclose a server-based approach, in which Java applications are executed at a server, which returns content to the user's web browser. *Id.* at 28–30 (citing Ex. 1003, 36 (describing “JavaServer Faces”), 76–77 (describing an application server as deploying the application), 52 (explaining that JavaServer Faces “helps you develop web applications using a server-side user interface component framework”); Ex. 1006, 45 (describing Java servlets as “server-side technologies that use the Java language”), 46 (illustrating a process in which a web server receives an http request and interprets code and sends back an HTML page to the user's browser)).

Petitioner responds by presenting evidence that, even if Anderson uses a server-side implementation, this does not require implementation using physically separate computers connected over a network. Reply 19–21. The Java application in Anderson can be run on either the same or a different machine from the one used to create the application using Creator. *Id.* at 19 (citing Ex. 1003, 77 (“You use localhost if you're running the application server on your own machine; otherwise, use the Internet address or host name where the server is running.”)). Moreover, Anderson expressly teaches an implementation in which a single computer contains both the client-side (e.g., web browser) and server-side (e.g., Java application executed by JVM) functionality. Reply 19–20; Pet. 63–64; Ex. 1002 ¶¶ 136, 152–153. For example, Anderson describes using the same machine – “localhost” – to run the GoogleSearch Java application and display its output in a web browser. Reply 20 (citing Ex. 1003, 303). Petitioner presents evidence that “the term ‘localhost’ refers to a well-known functionality in which a computer can issue a request to itself.” *Id.* (citing Ex. 1021

(Madisetti Reply Decl.) ¶ 48; Ex. 1022 (Microsoft Computer Dictionary (5th ed. 2002)), 7 (“**localhost** *n.* The name that is used to represent the same computer on which a TCP/IP message originates. An IP packet sent to localhost has the IP address 127.0.0.1 and does not actually go out to the Internet.”)). Thus, Petitioner argues, “[f]rom the standpoint of a software application that transmits and receives information, requests directed to ‘localhost’ (e.g. IP address ‘127.0.0.1’) behave no differently from requests directed to a remote computer over a network.” Reply 20 (citing Ex. 1021 ¶ 48 n.2).

Simply put, as Petitioner notes, the “localhost” technique disclosed in Anderson allows a single computer to execute both the “server-side” and “client side” functionality of a Java application—a common technique for development of networked applications to simplify implementation and testing. *See* Reply 20 (citing Ex. 1021 ¶ 48 & n.2). Petitioner further contends that “[c]ontrary to Patent Owner’s suggestions, there is nothing unusual about using the same computer in Anderson to execute both the ‘server-side’ Java servlets (via the JVM) and the ‘client-side’ web browser that accepts user input and displays search results.” *Id.* at 20–21. Patent Owner does not respond to this argument in its Sur-reply. *See generally* Sur-reply.

We agree with Petitioner. Anderson discloses using the “localhost” technique to run the Java application on the same machine used to create the application. *See* Ex. 1003, 77 (“You use localhost if you’re running the application server on your own machine; otherwise, use the Internet address or host name where the server is running.”); 303 (describing “localhost:24848” as “the application server host machine and port”);

Ex. 1021 ¶ 48 (Dr. Madisetti testifying that “Anderson describes using the same machine – ‘localhost’ – to run the GoogleSearch Java application and display its output in a web browser”). Dr. Madisetti testifies that “the term ‘localhost’ refers to a host name that [is] used with a request in order to refer back to the same (local) computer making the request; when a computer issues a request sent to ‘localhost,’ in other words, it is issuing a request to itself.” Ex. 1021 ¶ 48. Dr. Madisetti’s testimony relies on and is consistent with the Microsoft Computer Dictionary’s entry for the term “localhost,” which is as follows:

localhost *n.* The name that is used to represent the same computer on which a TCP/IP message originates. An IP packet sent to localhost has the IPR address 127.0.0.1 and does not actually go out to the Internet.

Ex. 1022, 316; *see* Ex. 1021 ¶ 48. Thus, Dr. Madisetti explains, “[t]he ‘localhost’ host name” allows “a single computer to perform the functions of both ‘client’ and ‘server’ for a particular application,” “is a standard technique for developing networked applications or for executing them locally,” and is “disclosed in Anderson” and “common and well-known to persons of ordinary skill in the art.” Ex. 1021 ¶ 48. Neither Patent Owner nor Dr. Almeroth provide persuasive contrary evidence showing why Petitioner’s arguments concerning the “localhost” technique are incorrect. *See* PO Resp.; Sur-reply; Ex. 2020.

(b) Patent Owner’s argument that Ambrose-Haynes teaches away from using Java and from a combination with Anderson

Patent Owner argues that Ambrose-Haynes teaches the use of the ColdFusion platform and thus teaches away from using Java. PO Resp. 35–38. Patent Owner asserts that “Ambrose-Haynes includes an entire chapter

that encourages the use of ColdFusion for web service applications that does not mention or encourage the use of Java for that purpose at all.” *Id.* at 37 (citing Ex. 1006, 29; Ex. 2034; Ex. 2020 ¶ 75). Patent Owner acknowledges that “Ambrose-Haynes does recognize that ColdFusion systems can leverage Java,” but argues that Ambrose-Haynes “does not do so in the context of web services, and teaches that Java should be used only for tasks that ‘ColdFusion was never intended to handle.’” *Id.* at 36 (citing Ex. 1006, 54).

Relatedly, Patent Owner argues that one of ordinary skill would not have combined Ambrose-Haynes with Anderson because “the references are focused on different and disparate platforms.” PO Resp. 39. According to Patent Owner, “ColdFusion, as discussed in Ambrose-Haynes, does not generate Java applications, but can merely be extended to utilize server-side Java.” *Id.* (citing Ex. 1006, 54). Additionally, Patent Owner argues again that Ambrose-Haynes teaches away from the proposed combination because it “explains at length why executing the application on a developer’s computer (i.e., client-side) is disadvantageous” because the “webmaster has no control over the software used to view the page” and “take-up is very slow”). *Id.* at 39–40 (citing Ex. 1006, 43–47).

Patent Owner further argues that Petitioner did not submit certain portions of Ambrose-Haynes into the record, including chapter 22 that “discusses many of the same features described in the Petition, including ‘Web services, SOAP, ‘XML,’ and ‘WDSL’” in the context of ColdFusion. PO Resp. 35–36. Additionally, Patent Owner contends that Dr. Madisetti was unable to confirm that those portions had been provided to him at his deposition. *Id.* at 35 (citing Ex. 2025, 173:25–174:18, 187:6–192:3).

Petitioner responds that Anderson and Ambrose-Haynes “are properly combinable,” and that Ambrose-Haynes does not teach away from the proposed combination. Reply 23–26. Petitioner argues that it cited Ambrose-Haynes for a “limited purpose,” namely to disclose “well-known characteristics of Java that Anderson takes for granted, including cross-platform capabilities enabled by use of the Java Virtual Machine (JVM).” *Id.* at 23 (citing Pet. 14). Petitioner states that it “is not relying on Ambrose-Haynes for a teaching or motivation to use Java for web services,” and thus “even if Patent Owner had identified something in Ambrose-Haynes discouraging use of Java for web services,” that “would be irrelevant because Anderson itself already discloses using Java for web services.” *Id.* (citing Ex. 1021 ¶ 53).

Petitioner also argues that Ambrose-Haynes does not discourage or teach away from using Java but “merely recognizes Java and ColdFusion as alternatives to one another.” Reply 23–24 (citing Ex. 1021 ¶¶ 52–53). Petitioner contends that Patent Owner has not identified anything in Ambrose-Haynes that criticizes, discredits, or otherwise discourages the use of Java for web services. *Id.* at 24. To the contrary, Petitioner argues, Ambrose-Haynes states that Java can be used for tasks that “ColdFusion was never intended to handle,” thereby “praising Java and encouraging its use to extend ColdFusion’s capabilities.” *Id.* (citing Ex. 1021 ¶ 52; Ex. 1006, 54, 69).

As for Patent Owner’s argument that only a portion of Ambrose-Haynes was included as an exhibit to the Petition, Petitioner responds that its submission “was attributable to the limited purpose for which [Ambrose-

Haynes] was cited,” and that Patent Owner could have included other relevant sections if it chose to do so. Reply 24–25.

We agree with Petitioner that Ambrose-Haynes does not teach away from the use of Java, and that Petitioner’s reliance on Anderson and Ambrose-Haynes was appropriate. As we explained earlier, Petitioner offers the secondary references, including Ambrose-Haynes, more to provide details that are omitted from Anderson than as the basis for any modification of Anderson. More specifically, the Petition explains that the combination relies on Ambrose-Haynes “for its description of well-known characteristics of Java that Anderson takes for granted, including cross-platform capabilities enabled by use of the Java Virtual Machine (JVM).” Pet. 14 (citing Ex. 1002 ¶¶ 59–61); *see* Pet. Reply 23 (explaining that Ambrose-Haynes was cited for the “limited purpose” of disclosing “well-known characteristics of Java that Anderson takes for granted”). Similarly, Dr. Madisetti explains that he is “not relying on Ambrose-Haynes for a teaching or motivation to use Java for web services or to modify Anderson in any way,” but instead “only point[s] to Ambrose-Haynes because it provides additional detail about the JVM mentioned in Anderson,” which “would have already been part of the basic knowledge possessed by an ordinarily skilled artisan.” Ex. 1021 ¶ 53. Thus, there is no modification of Anderson for Ambrose-Haynes to teach away from.

Moreover, we find that Ambrose-Haynes does not teach away from using Java in Anderson’s system. “[A] reference may be said to teach away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant.” *In re*

Mouttet, 686 F.3d 1332, 1333–134 (Fed. Cir. 2012). “Thus, the ‘mere disclosure of alternative designs does not teach away.’” *Id.* at 1334. “[J]ust because better alternatives exist in the prior art,” the Federal Circuit explained, “does not mean that an inferior combination is inapt for obviousness purposes.” *Id.*

The portions of Ambrose-Haynes that Patent Owner relies on merely discuss ColdFusion as an alternative to Java, and do not discourage or discredit the use of Java. For example, the fact that Ambrose-Hayne’s table of contents for Chapter 22 discusses ColdFusion and does not mention Java is not a discouragement or discrediting of the use of Java for web services. *See* Ex. 1006, 29; PO Resp. 36–37. Simply because Chapter 22 encourages the use of ColdFusion does not mean it teaches away from Java.

Patent Owner’s reliance on Dr. Almeroth’s testimony is also unavailing. Dr. Almeroth testifies that Chapter 22 of Ambrose-Haynes encourages the use of ColdFusion, citing various statements from Chapter 22. Ex. 2020 ¶ 75 (citing Ex. 2034, 2 (“We will use a combination of ColdFusion and ASP to handle *all communication*.”), 10 (“[W]e will look at how ColdFusion can issue SOAP Requests.”), 13 (describing how to “use ColdFusion to read the contents of the SOAP Response and act upon it”), 27 (describing that “ColdFusion Server . . . sends a request to a web server over HTTP”), 29 (explaining how to “interact with this .NET web service via the ColdFusion language”)). These statements, however, merely encourage the use of ColdFusion; they do not discourage or discredit the use of Java. Similarly, Dr. Almeroth points to Chapter 25’s statement that the ColdFusion platform includes “an easy to use, flexible, web application development tool,” but again this merely encourages the use of ColdFusion

without discouraging or disparaging Java. Ex. 2020 ¶ 75 (citing Ex. 1006, 54). Thus, we find that Dr. Almeroth’s conclusion that Ambrose-Haynes teaches away from using Java in the context of web services is not supported by the portions of the reference he cites. *Id.* ¶¶ 74–75. Accordingly, we give that testimony little weight. Additionally, we find that Dr. Madisetti’s testimony that “Ambrose-Haynes does not teach away from using Java” and “simply recognizes that Java and ColdFusion can be alternatives to each other” to be credible and supported by the disclosures of the reference. Ex. 1021 ¶ 52.

Finally, we are not persuaded by Patent Owner’s argument that Petitioner only submitted part of the Ambrose-Haynes book into the record, and that Dr. Madisetti was not able to confirm whether certain portions of Ambrose-Haynes that he did not rely on were provided to him. *See* PO Resp. 35–36. As Petitioner points out, Patent Owner had the full Ambrose-Haynes book and could have submitted into the record and raised arguments concerning any portions of the book it wished. Reply 24–25. We do not find that Dr. Madisetti’s failure to read the entire Ambrose-Haynes book undercuts his reliance on the book’s discussion of Java, which he used only to provide details about Java’s operation not expressly set forth in Anderson. As Dr. Madisetti explains, his “analysis relies on Ambrose-Haynes for the limited purposes of describing the JVM and how it is loaded and used for Java applications.” Ex. 1021 ¶ 54.

Consequently, based on the record before us, we find that Ambrose-Haynes does not teach away from using Java for web services or from using its disclosure to provide further details on Java and the JVM not expressly set forth in Anderson.

(c) Patent Owner's argument that the references teach different and incompatible platforms

Patent Owner argues that Petitioner's references "are directed toward different and incompatible systems, platforms, and languages." PO Resp. 41 (citing Ex. 2020 ¶¶ 79–82). Specifically, Patent Owner argues that Anderson "describes a tool for generating server-side Java applications," Bowers "describes integrating a desktop database application, FileMaker Pro, with web services," Jacobs "describes the use of XML with a Macromedia Flash video player," and Ambrose-Hayes "describes a proprietary server-side platform called ColdFusion." *Id.* According to Patent Owner, "Macromedia Flash, FileMaker Pro, and ColdFusion are disparate, monolithic software platforms based on competing technologies that were never intended to be used in combination," and one of ordinary skill "would never have combined such different references without the benefit of hindsight." *Id.* at 42 (citing Ex. 2020 ¶¶ 80–82).

Petitioner responds that Patent Owner fails to explain how the alleged differences in the systems of the various references have any bearing on how they are used in the proposed combination. Pet. Reply 23. For example, Petitioner argues that it did not cite the Bowers and Jacobs references for their disclosures about FileMaker and Macromedia Flash, respectively, but "instead cited them for their detailed descriptions about the Google web service described in Anderson." *Id.*

We agree with Petitioner. As discussed above, Petitioner cites Ambrose-Haynes to provide additional details about Java not expressly set forth in Anderson. *See supra* § II.E.1.a(5)(b). The Petition states that it relies on Bowers and Jacobs "to provide additional details about the same

Google search web services described in Anderson,” including “the specific content and format of the actual WSDL file for the Google web service, and the format and content of actual input and output messages sent and received using that service.” Pet. 14 (citing Ex. 1002 ¶¶ 53–54, 56–57). The Petition does not rely on Bowers and Jacobs for their disclosures about FileMaker and Macromedia Flash. *Id.* Moreover, to the extent Patent Owner is arguing that it would not have been obvious to physically combine the references or bodily incorporate them into each other, Dr. Madisetti explains that his “analysis is not based on physical combinability or bodily incorporation,” but that he merely relied on Bowers and Jacobs to “provide detail that Anderson takes for granted,” rather than “as a basis for any modification of Anderson.” Ex. 1021 ¶ 59. We find that Petitioner properly relies on the additional details in Bowers and Jacobs about the Google search web services described in Anderson, and that the disclosures about FileMaker and Macromedia Flash in Bowers and Jacobs would not have detracted from the relevance of their disclosures concerning the Google search web services that Anderson describes.

(6) *Objective Indicia*

In separate briefing, authorized by the Board, Patent Owner contends that patentability is supported by objective indicia of nonobviousness. PO Obj. Evid. Br. 1–5. In particular, Patent Owner contends that a “federal jury trial recently supplied objective evidence of nonobviousness” as to the ’287 patent, and “two patents sharing the same disclosure,” wherein the jury “found the broadest challenged claim of this patent (claim 1) to be infringed and nonobvious and awarded \$40 million for infringement of the three patents’ claims.” *Id.* at 1; *see* Ex. 2041 (jury verdict in *Shopify Inc. v.*

Express Mobile Inc. (CA No. 19-439-RGA (D. Del. Aug. 31, 2022) (hereafter “*Shopify* Verdict”)). Patent Owner argues that the *Shopify* Verdict evidences both “commercial success” and a “failed attempt to invalidate the claims.” PO Obj. Evid. Br. 1.

Petitioner responds by noting, first, that Patent Owner’s evidence of objective indicia (secondary considerations) is untimely because Patent Owner failed to present such evidence in its Patent Owner Response. Pet. Obj. Evid. Resp. Br. 1. Although Patent Owner filed its Response on June 23, 2022—over two months before the *Shopify* Verdict—Petitioner contends that the *Shopify* litigation began in March 2019, and “Patent Owner fully developed its underlying evidence and theory relating to secondary considerations more than two years ago.” *Id.* Petitioner insists that Patent Owner could have presented evidence of secondary considerations pertaining to the *Shopify* products earlier, and its failure to do so “has deprived Petitioner of a full and fair opportunity to respond.” *Id.*

In view of our ultimate determination of the relevance of the *Shopify* Verdict, discussed further below, we decline to find that Patent Owner’s presenting this evidence in supplemental briefing with Board approval, roughly two months after the issuance of the verdict, was untimely.

(a) Legal Standards

Objective indicia of nonobviousness may include long-felt but unsolved need, failure of others, unexpected results, commercial success, copying, licensing, industry praise, and expert skepticism. *Mintz v. Dietz & Watson, Inc.*, 679 F.3d 1372, 1379 (Fed. Cir. 2012). “[O]bjective indicia may often be the most probative and cogent evidence of nonobviousness in the record,” and “help turn back the clock and place the claims in the context

that led to their invention.” *Id.* at 1378. Evidence of objective indicia of nonobviousness “must always when present be considered en route to a determination of obviousness.” *Transocean Offshore Deepwater Drilling, Inc. v. Maersk Drilling USA, Inc.*, 699 F.3d 1340, 1349 (Fed. Cir. 2012); *see also Apple Inc. v. Samsung Elecs. Co.*, 839 F.3d 1034, 1048 (Fed. Cir. 2016) (en banc).

Objective indicia of nonobviousness are “only relevant to the obviousness inquiry ‘if there is a nexus between the claimed invention and the [objective indicia of nonobviousness].’” *In re Affinity Labs of Tex., LLC*, 856 F.3d 883, 901 (Fed. Cir. 2017) (quoting *Ormco Corp. v. Align Tech., Inc.*, 463 F.3d 1299, 1312 (Fed. Cir. 2006)). For objective indicia of nonobviousness to be accorded substantial weight, their proponent must establish a nexus between the evidence and the merits of the claimed invention. *ClassCo, Inc. v. Apple Inc.*, 838 F.3d 1214, 1220 (Fed. Cir. 2016).

As the Federal Circuit has explained, “a patentee is entitled to a rebuttable presumption of nexus between the asserted evidence of secondary considerations and a patent claim if the patentee shows that the asserted evidence is tied to a specific product and that the product ‘is the invention disclosed and claimed.’” *Fox Factory, Inc. v. SRAM, LLC*, 944 F.3d 1366, 1373 (Fed. Cir. 2019) (quoting *Demaco Corp. v. F. Von Langsdorff Licensing Ltd.*, 851 F.2d 1387, 1392 (Fed. Cir. 1988)). On the other hand, the patentee is not entitled to a presumption of nexus if the patented invention is only a component of a commercially successful machine or process. *Id.* Once “the patentee has presented a *prima facie* case of nexus, the burden of coming forward with evidence in rebuttal shifts to the

challenger . . . to adduce evidence to show that the commercial success was due to extraneous factors other than the patented invention.” *Demaco*, 851 F.2d at 1392–93.

(b) Commercial Success

Patent Owner contends that “commercial success” of the claimed invention is demonstrated by the *Shopify* Verdict’s award of \$40 million in damages to Patent Owner for infringement by Shopify of the ’287 patent along with two related patents. PO Obj. Evid. Br. 3 (citing Ex. 2042). Patent Owner further contends that “[t]here is a ‘nexus for the objective consideration[]’ of commercial success between the verdict and the claims” because “the court found that the products ‘incorporate the patented technology’ and awarded substantial damages for that incorporation.” *Id.* Patent Owner asserts that the \$40 million award by the jury was “attributed solely to the patented features.” *Id.*

Petitioner responds that Patent Owner has “failed to provide the factual and evidentiary context needed to assess the probative value (if any) of the Shopify verdict,” and in particular “offers no information about the identity of the accused Shopify products on which it relies for commercial success, let alone how those products worked or the total revenue attributable to them.” Pet. Obj. Evid. Resp. Br. 2. Thus, argues Petitioner, Patent Owner has failed to establish “any causal nexus between the claims and the success of Shopify’s products.” *Id.* at 3.

We agree with Petitioner that Patent Owner has not shown a nexus between the argued commercial success and the merits of the claimed invention. Patent Owner’s evidence establishes that Shopify infringed three claims of the ’287 patent (as well as claims of two other patents), but no

products are identified on the verdict form. *See* Ex. 2041 (jury verdict finding infringement). Moreover, because there is no evidence that the alleged commercial success is due to the elements of the '287 patent—either individually or as a whole—Patent Owner has not established the required nexus. Establishing that a product infringes a patent is not enough to show a nexus. *See Fox Factory*, 944 F.3d at 1377 (holding that a *prima facie* case of nexus cannot be made by simply showing that “the patent claims broadly cover the product that is the subject of the evidence of secondary considerations”).

Even if a nexus had been shown, there is insufficient evidence to show that the infringing Shopify products were a commercial success. The jury verdict form simply asks whether Shopify “infringes the identified claim” of the three asserted patents (including the '287 patent) and then asks for a total amount of damages for the infringement. *See* Ex. 2041. There is no evidence—such as a large market share—establishing that the unidentified Shopify products underlying the jury’s verdict were a commercial success. *See, e.g., In re Applied Materials, Inc.*, 692 F.3d 1289, 1300 (Fed. Cir. 2012) (“An important component of the commercial success inquiry in the present case is determining whether Applied had a significant market share.”).

Patent Owner cites no authority that a broad jury verdict of infringement, involving unidentified products of a non-party, is relevant as objective indicia of nonobviousness in this proceeding. To the contrary, as Petitioner aptly notes, the Board has repeatedly found such out-of-context jury verdicts to have minimal value in establishing objective indicia of nonobviousness. *See, e.g., Mangrove Partners v. VirnetX Inc.*,

IPR2015-01047, Paper 122 at 25 (PTAB July 14, 2020); *Ingenico Inc. v. IOENGINE, LLC*, IPR2019-00929, Paper 53 at 93–94 (PTAB Sept. 21, 2020).

Patent Owner relies upon *RTI Surgical, Inc. v. LifeNet Health*, IPR2019-00571, Paper 79 at 46–47 (PTAB Aug. 4, 2020), but the IPR petitioner there was the same party against whom the verdict was entered. And unlike Patent Owner here, the patent owner in *RTI* did not simply rely on the jury’s verdict; rather, the patent owner submitted its underlying evidence of secondary considerations with its Patent Owner Response—which is what the Board relied upon. *See id.* at 46–47. Similarly, another decision relied upon by Patent Owner—*WBIP, LLC v. Kohler Co.*, 829 F.3d 1317 (Fed. Cir. 2016)—involved an appeal of a district court trial involving issues of willful infringement, invalidity, and damages against the same defendant. The Federal Circuit did not rely on the mere existence of an infringement verdict as evidence of commercial success, but rather determined that extensive underlying evidence in the record supported a finding of commercial success and nexus. *Id.* at 1329–31.

For the foregoing reasons, we determine that, at best, Patent Owner’s evidence of commercial success is weak.

(c) Failed Attempt by Another to Show Invalidity

Patent Owner also contends that the *Shopify* Verdict evidences nonobviousness because the jury rejected Shopify’s obviousness challenge in that case. PO Obj. Evid. Br. 4. Patent Owner posits that “[t]he fact ‘that the patent in suit had successfully withstood previous validity challenges in other proceedings’ presents ‘a clear case supporting the validity of the patent.’” *Id.* (citing *Amazon.com, Inc. v. Barnesandnoble.com, Inc.*, 239

F.3d 1343, 1359 (Fed. Cir. 2001)). Patent Owner also cites *Custom Accessories, Inc. v. Jeffrey-Allan Indus.*, 807 F.2d 955, 961 (Fed. Cir. 1986), for the proposition that “previous validity challenges that the claims survived are ‘evidence the court must consider’ in favor of validity.” *Id.*

Petitioner points out that the validity challenge raised by Shopify involved different prior art than Petitioner presents here. Pet. Obj. Evid. Resp. Br. 5. Patent Owner attempts to put a positive spin on this point by stating that Shopify “preferred to not even attempt to present the grounds in *this* case to the jury.” PO Obj. Evid. Br. 5.

We conclude that the *Shopify* Verdict is of little, if any, probative value in terms of the jury’s rejection of an obviousness challenge in that case. As Petitioner points out, and Patent Owner acknowledges, the obviousness challenge underlying the *Shopify* Verdict did not involve the same prior art references as Petitioner presents here. Therefore, the jury did not find Patent Owner’s claims patentable over the combination that we have addressed herein. Whether Shopify actually considered, but declined to present, the same obviousness challenge as Petitioner now presents—a fact implied by Patent Owner but one that we cannot assess on the present record—is, even if true, immaterial.

The cases cited by Patent Owner do not support the proposition that a jury verdict of nonobviousness over different prior art is evidence that we must consider here. The *Custom Accessories* decision involved a previous reexamination proceeding before the Patent Office in which “much the same art” was addressed. *Custom Accessories*, 807 F.2d at 961. The *Amazon.com* decision involved a review of a grant of a preliminary injunction against the accused infringer. *Amazon.com*, 239 F.3d at 1347. In assessing that grant,

the Federal Circuit observed, generally, that “[w]hen moving for the extraordinary relief of a preliminary injunction, the patentee need not establish the validity of a patent beyond question,” but admonished that the patentee must “present a clear case supporting the validity of the patent in suit.” *Id.* at 1359. The Federal Circuit also observed that “[s]uch a case might be supported, for example, by showing that the patent in suit had successfully withstood previous validity challenges in other proceedings.” *Id.* In so stating, however, the Federal Circuit did not indicate the relevance of a previous validity challenge based on completely different prior art, nor was any sort of previous validity challenge even before the court in that case. Rather, the Federal Circuit ultimately vacated the grant of a preliminary injunction because the district court, in granting that relief, had failed to properly assess the very prior art raised by the defendant in that case. *See id.* at 1359–66.

We conclude that the *Shopify* Verdict, in which the jury rejected an obviousness challenge over different prior art, is entitled to little, if any weight, as objective evidence of nonobviousness here.

(7) *Claim 1: Conclusion*

After having analyzed the entirety of the trial record and assigning appropriate weight to the cited supporting evidence, we determine Petitioner has shown by a preponderance of the evidence that, at the time of the filing of the ’287 patent, one of ordinary skill would have been motivated to combine the teachings of Anderson, Bowers, Jacobs, Ambrose-Haynes, and Geary in the manner recited in claim 1.

We have considered Patent Owner’s proffered evidence of secondary considerations of commercial success and prior determination of validity.

For the reasons expressed above, we give little weight to that evidence, and find that it does not overcome Petitioner’s persuasive showing of obviousness.

b. Claims 2, 5–7, and 12

Petitioner additionally presents evidence that claims 2, 5–7, and 12, which depend from claim 1, are unpatentable over the combined teachings of Anderson, Bowers, Jacobs, Ambrose-Haynes, and Geary. Pet. 78–81. Patent Owner does not present arguments or evidence particularly directed to these claims, but relies on its arguments presented in connection with claim 1, which we address above.

Claim 2 depends from claim 1, and recites “where said registry includes definitions of input and output related to said web service.” Ex. 1001, 38:38–39. Petitioner presents evidence that the GoogleSearch.wsdl file in Anderson corresponds to “said registry” and includes definitions for input and output messages for the doGoogleSearch, doSpellingSuggestion, and doGetCachedPage methods. Pet. 78 (citing Ex. 1002 ¶¶ 154–155, 160–162, 166); *see* Ex. 1004, 100,102 (depicting input and output messages for doGoogleSearch). We agree that the cited evidence supports Petitioner’s contentions.

Claim 5 depends from claim 1, and recites “where said UI object is an input field for a web service.” Ex. 1001, 38:45–46. Petitioner presents evidence that “said UI object” corresponds to a Text Field in Anderson “for providing a search query for the doGoogleSearch method of the GoogleSearch web service,” which Petitioner contends is an “input field for a web service.” Pet. 78 (citing Ex. 1003, 286–287; Ex. 1002 ¶¶ 118–120, 167). We agree that the cited evidence supports Petitioner’s contentions.

Claim 6 depends from claim 1, and recites “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.” Ex. 1001, 38:47–50. Petitioner presents evidence that “said UI object” in Anderson “corresponds in one embodiment to a Text Field for providing a search query for the doGoogleSearch method of the GoogleSearch web service,” which Petitioner contends is “an input field ... where said input field includes a text field.” Pet. 78–79 (citing Ex. 1002 ¶ 168). Petitioner further presents evidence that this UI object in Anderson is also “usable to obtain said web component,” i.e. the doGoogleSearch method, “because the text field is integral to obtaining the functionality provided by that method.” *Id.* at 79. In order to obtain search results through the doGoogleSearch method (“said web component”), both the user and the web application must use a text field (“said UI object”) to obtain the search query and provide that query to the web service. *Id.* (citing Ex. 1002 ¶ 169). We agree that the cited evidence supports Petitioner’s contentions.

Claim 7 depends from claim 1, and recites “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.” Ex. 1001, 38:51–55. Petitioner presents evidence that Anderson’s doGoogleSearch method corresponds to the claimed “said web component” and qualifies as “an output of a web service” “because it provides output of the GoogleSearch web service, *i.e.* one or more search results based on an input

search query.” Pet. 79 (citing Ex. 1002 ¶ 170). We agree that the cited evidence supports Petitioner’s contentions.

Claim 12 depends from claim 1, and recites “wherein said defined UI object corresponds to a widget.” Ex. 1001, 39:8–9. Petitioner first notes that “[t]he ’287 patent does not define ‘widget,’” but presents testimony from Dr. Madisetti, along with an excerpt from a computing dictionary, that “the term is commonly used as a buzzword referring to a user interface object that can be added to an application, with which a user can interact.” Pet. 79 (citing Ex. 1002 ¶ 171; Ex. 1012, 7 (New Penguin Dictionary of Computing (2001))). Petitioner then contends that “[n]othing in the ’287 patent appears to restrict the type of user interface controls that can qualify as a ‘widget.’” *Id.* at 79–80. Petitioner then presents evidence that Anderson teaches that the UI objects identified for claim 1 (*see supra* II.E.1.a) “individually and collectively, qualify as a widget.” *Id.* at 80 (citing Ex. 1003, 287, Fig. 10-1). In particular, Petitioner presents evidence that, in Anderson, both the Text Field and the Grid Panel (individually and collectively) qualify as a “widget” because “they provide user interface elements for interacting with the user by receiving user input (i.e. query string) and presenting output to the user (i.e. search result information).” *Id.* at 80–81 (citing Ex. 1002 ¶¶ 171–172). Additionally, Anderson’s description of adding these UI objects to a web application is consistent with the description in the ’287 patent specification. *Id.* at 81 (citing Ex. 1002 ¶ 172; *see also* Ex. 1001, 25:9–15, 25:31–33, Fig. 6B; Ex. 1003, 288–289, 293–295). Petitioner further contends that “it would have been obvious to implement these UI objects through reusable Java classes and in accordance with known object-oriented programming techniques to enable reusability

with other applications.” *Id.* (citing Ex. 1002 ¶¶ 172, 101, 102, 111). We agree that the cited evidence supports Petitioner’s contentions.

Petitioner’s contentions, summarized above, are persuasive. In addition, as explained above in the context of claim 1, we have considered Patent Owner’s proffered evidence of secondary considerations of commercial success and prior determination of validity. For the reasons expressed above, we give little weight to that evidence, and find that it does not overcome Petitioner’s persuasive showing of obviousness.

After having analyzed the entirety of the trial record and assigning appropriate weight to the cited supporting evidence, we conclude that the subject matter of claims 2, 5–7, and 12 would have been obvious to a person of ordinary skill in the art based on the combined teachings of Anderson, Bowers, Jacobs, Ambrose-Haynes, and Geary.

c. Ground 1: Conclusion

For the foregoing reasons, we determine that Petitioner has established by a preponderance of the evidence that claims 1, 2, 5–7, and 12 are unpatentable over the combination of Anderson, Bowers, Jacobs, Ambrose-Haynes, and Geary.

2. Ground 2: Claim 11 over Anderson, Bowers, Jacobs, Ambrose-Haynes, Geary, and NFS Administration

In Ground 2, Petitioner presents arguments and evidence that dependent claim 11 is unpatentable over the combination of Anderson, Bowers, Jacobs, Ambrose-Haynes, Geary, and NFS Administration. Pet. 81–85.

Claim 11 depends from claim 1, and recites “where said code is provided over said network.” Ex. 1001, 39:6–7.

Petitioner contends that “[i]t was well-known that users could store and retrieve any type of computer file (including files storing the Player and Application of claim 1) using remote storage provided over a network, such as the Internet.” Pet. 81 (citing Ex. 1002 ¶¶ 174, 177). Petitioner relies in particular on NFS Administration as providing this teaching. *Id.* at 81–83. Relying in part on the testimony of Dr. Madisetti, Petitioner also provides arguments and evidence regarding a motivation to combine the art cited in Ground 1 with NFS Administration to add the ability “to obtain the code for the Application and the Player over the Internet using a distributed file system such as NFS.” *Id.* at 83–85 (citing Ex. 1002 ¶¶ 178, 180–184).

Petitioner’s contentions are persuasive and supported by the cited teachings. NFS Administration explains, for example, that “NFS file sharing is used to make resources on a local system available to remote systems and, conversely, to access resources on remote systems from a local system. Using NFS, it is possible to share individual files, file hierarchies, and entire file systems across a network.” Ex. 1007, 10.

Patent Owner’s arguments against Petitioner’s showing are limited to those we have addressed above in connection with Ground 1. Patent Owner does not present separate arguments particularly against Petitioner’s showing on this ground.

Petitioner’s contentions, summarized above, are persuasive. In addition, as explained above in the context of claim 1, we have considered Patent Owner’s proffered evidence of secondary considerations of commercial success and prior determination of validity. For the reasons expressed above, we give little weight to that evidence, and find that it does not overcome Petitioner’s persuasive showing of obviousness.

After having analyzed the entirety of the trial record and assigning appropriate weight to the cited supporting evidence, we conclude that the subject matter of claim 11 would have been obvious to a person of ordinary skill in the art based on the combined teachings of Anderson, Bowers, Jacobs, Ambrose-Haynes, Geary, and NFS Administration.

III. PATENT OWNER'S OBJECTIONS TO EXHIBITS

Patent Owner filed objections to evidence in response to Petitioner's Reply. Paper 20. However, those objections were not subsequently preserved with a motion to exclude, and thus no relief can be granted concerning those objections. *See* 37 C.F.R. § 42.64(c) ("A motion to exclude evidence must be filed to preserve any objection."). Patent Owner's objections are overruled.

IV. CONCLUSION

For the foregoing reasons, we determine Petitioner has demonstrated by a preponderance of the evidence that claims 1, 2, 5–7, 11, and 12 of the '287 patent are unpatentable.

V. ORDER

For the reasons given, it is:

ORDERED that claims 1, 2, 5–7, 11, and 12 of U.S. Patent No. 9,471,287 B2 are held unpatentable under 35 U.S.C. § 103 as obvious; and

FURTHER ORDERED that, because this is a Final Written Decision, parties to the proceeding seeking judicial review of this decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

In summary:

Claim(s)	35 U.S.C. §	References / Basis	Claim(s) Shown Unpatentable	Claim(s) Not Shown Unpatentable
1, 2, 5–7, 12	103(a)	Anderson, Bowers, Jacobs, Ambrose-Haynes, Geary	1, 2, 5–7, 12	
11	103(a)	Anderson, Bowers, Jacobs, Ambrose-Haynes, Geary, NFS Administration	11	
Overall Outcome			1, 2, 5–7, 11, 12	

IPR2021-01456
Patent 9,471,287 B2

PETITIONER:

Heidi L. Keefe
Mark Weinstein
Phillip Morton
Andrew Mace
Chih Yun Wu
Dustin Knight
COOLEY LLP
hkeefe@cooley.com
pmorton@cooley.com
amace@cooley.com
swu@cooley.com
dknight@cooley.com

PATENT OWNER:

Sal Lim
David Alberti
Russell Tonkovich
Hong S. Lin
Aidan Brewster
KRAMER DAY ALBERTI LIM
TONKOVICH & BELLOLI LLP
slim@feinday.com
dalberti@feinday.com
rtonkovich@feinday.com
hlin@feinday.com

Kenneth J. Weatherwax
Bridget Smith
LOWENSTEIN & WEATHERWAX LLP
weatherwax@lowensteinweatherwax.com
smith@lowensteinweatherwax.com