

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CHERWELL SOFTWARE, LLC,
Petitioner,

v.

BMC SOFTWARE, INC.,
Patent Owner.

Case IPR2018-00982
Patent 6,895,586 B1

Before JUSTIN T. ARBES, JOHN A. HUDALLA, and
CHRISTOPHER L. OGDEN, *Administrative Patent Judges*.

HUDALLA, *Administrative Patent Judge*.

DECISION
Denying Institution of *Inter Partes* Review
35 U.S.C. § 314(a)

Petitioner, Cherwell Software, LLC (“Petitioner”), filed a Petition (Paper 3, “Pet.”) requesting an *inter partes* review of claims 1–35 of U.S. Patent No. 6,895,586 B1 (Ex. 1001, “the ’586 patent”). Petitioner filed a Declaration of Edward A. Fox, Ph.D. (Ex. 1003) with its Petition. Patent Owner, BMC Software, Inc. (“Patent Owner”), filed a Preliminary Response (Paper 8, “Prelim. Resp.”).

We have authority to determine whether to institute an *inter partes* review. See 35 U.S.C. § 314(b); 37 C.F.R. § 42.4(a). Under 35 U.S.C. § 314(a), we may not authorize an *inter partes* review unless the information in the petition and the preliminary response “shows that there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.” For the reasons that follow, we do not institute an *inter partes* review.

I. BACKGROUND

A. *Related Proceedings*

The parties identify the following proceedings related to the ’586 patent (Pet. 2–3; Paper 6, 1):

BMC Software, Inc. v. Cherwell Software, LLC, No. 17-cv-03127-STV (D. Colo. filed Dec. 22, 2017). This case was transferred on December 21, 2017, from the Eastern District of Virginia, where it was filed as Case No. 17-cv-0174-LO-TCB on September 22, 2017;

ServiceNow, Inc. v. BMC Software, Inc., Case IPR2015-01329 (institution denied Dec. 18, 2015 (Paper 10)) (“the ServiceNow IPR”);

BMC Software, Inc. v. Cherwell Software, LLC & FireScope, Inc., No. 3:17-cv-01369-N (N.D. Tex. filed May 22, 2017);

BMC Software, Inc. v. ServiceNow, Inc., No. 14-cv-00903-JRG (E.D. Tex. filed Sept. 23, 2014) (“the 2014 Texas litigation”); and

BMC Software, Inc. v. ServiceNow, Inc., No. 16-cv-00132-JRG (E.D. Tex. filed Feb. 12, 2016).

B. The '586 patent

The '586 patent, titled “Enterprise Management System and Method which Includes a Common Enterprise-Wide Namespace and Prototype-Based Hierarchical Inheritance,” issued from an application filed August 30, 2000. Ex. 1001, [22], [54]. The '586 patent is directed to a method for managing one or more networked computer systems (an “enterprise”). *Id.* at 1:21–22. The method employs a “hierarchical namespace” for efficient referencing and retrieval of information to manage the enterprise. *Id.* at Abstract, 1:50–52. The hierarchical namespace is a logical arrangement of “objects” in a hierarchical, uniquely addressable system of object names. *Id.* at 1:54–60, 2:12–13. The objects typically relate to the “monitoring and analysis activities of the enterprise,” and therefore relate to the hardware and software components in the enterprise. *Id.* at 9:22–26. The Internet is one example of an enterprise namespace in which many individual computer systems are linked together through the use of a hierarchical name structure and addressing system, where “every server name is different from every other server name.” *Id.* at 1:61–67.

Figure 6, which is reproduced below, illustrates the organization of information in a hierarchical namespace.

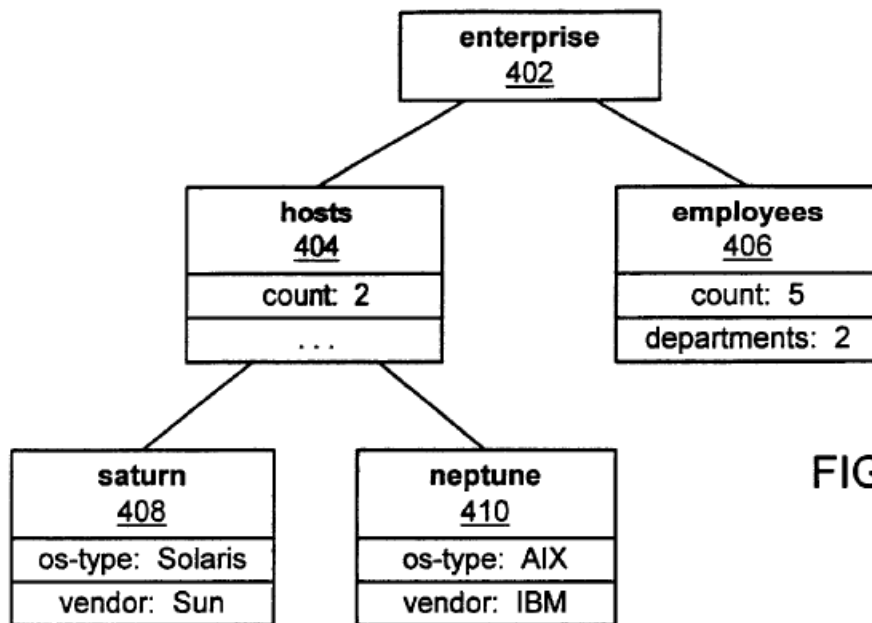


FIG. 6

In Figure 6, root object 402 at the top of the hierarchy is named “enterprise,” and it has two child objects 404, 406 named “hosts” and “employees,” respectively. *Id.* at 13:52–56. “[H]osts” 404 has two child objects 408, 410 named “saturn” and “neptune,” respectively. *Id.* at 13:56–58. The vendor for each host is a so-called “object attribute” that is uniquely addressed by name, for example, by the path “hosts/saturn vendor.” *Id.* at 14:5–7.

The method of the ’586 patent manages the enterprise by sharing objects with one or more applications and/or computer systems in the enterprise. *Id.* at 9:28–32. In particular, the method for sharing objects uses a technique called “dynamic inheritance.” *Id.* at 14:41–15:2. Dynamic inheritance includes the ability of one object “to derive attributes, values, and/or children from another object, where the attributes, values, and/or children may change over time.” *Id.* at 14:57–60. Dynamic inheritance ensures that changes to object data, which reflect changes in the components

of the enterprise, are shared among the appropriate components in the hierarchical namespace. *Id.* at 14:52–15:2, Fig. 7.

Figure 7 illustrates the technique of dynamic inheritance, and is reproduced below.

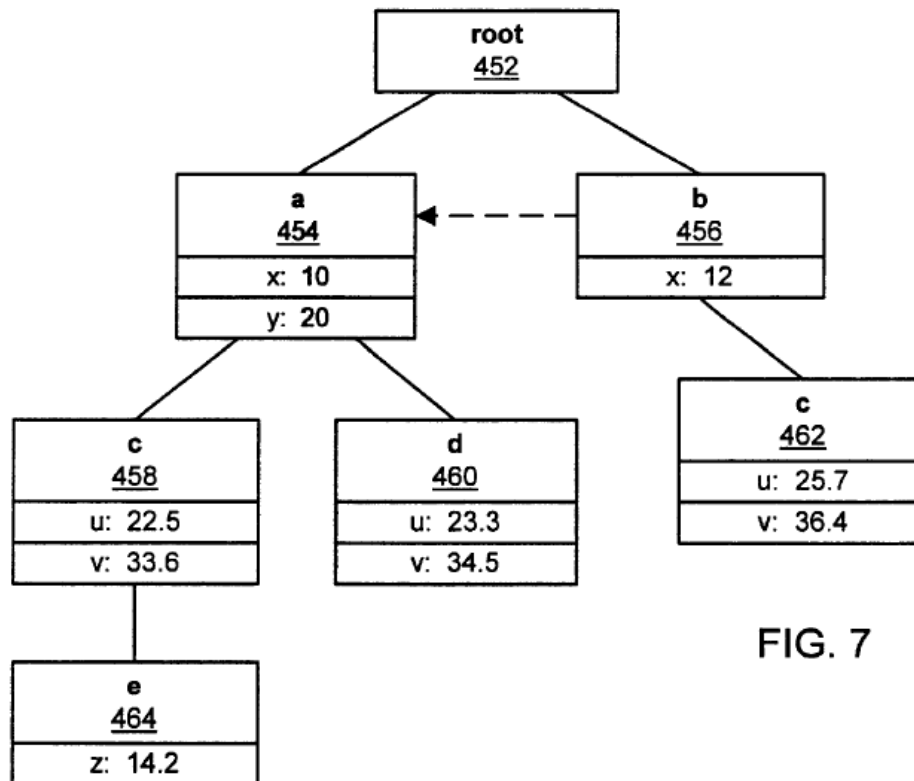


FIG. 7

In Figure 7, root object 452 at the top of the hierarchy has child objects 454, 456 named “a” and “b,” respectively, which are connected by a dynamic inheritance link represented by the dashed arrow pointing from “b” to “a.” *Id.* at 14:51–54. In the example of Figure 7, object “b” will dynamically inherit certain attributes, values, and/or children from object “a.” *Id.* at 14:55–15:2. Because object “b,” as compared to object “a,” already has an attribute named “x” of its own and a child object named “c” of its own, object “b” will inherit only attribute “y” and child object “d” from object “a.” *Id.* The ’586 patent styles object “a” as a “prototype” object, “from

which attributes, values, and/or children are dynamically inherited by another object,” and object “b” as an “instance” object that “dynamically inherits attributes, values, and/or children from another object in the namespace.” *Id.* at 14:44–49.

C. Illustrative Claim

Of the challenged claims, claims 1, 8, 14, 21, and 28 of the ’586 patent are independent. Claims 2–7 depend from claim 1; claims 9–13 depend from claim 8; claims 15–20 depend from claim 14; claims 22–27 depend from claim 21; and claims 29–35 depend from claim 28. Claim 1 is illustrative of the challenged claims and recites:

1. A method for managing an enterprise, wherein the enterprise comprises one or more networked computer systems, the method comprising:
 - providing a hierarchical namespace;
 - adding a plurality of objects to the namespace, wherein the objects relate to software and hardware of the one or more computer systems;
 - sharing the plurality of objects with a plurality of the one or more computer system, wherein at least one of the objects is a prototype and at least one of the objects is an instance, wherein the instance dynamically inherits traits from the prototype; and wherein the values of the traits inherited from the prototype change dynamically.

D. Prior Art

Petitioner relies on the following prior art:

U.S. Patent No. 6,862,736 B2 to Hudis et al., filed Feb. 13, 2002, continuation of application filed Feb. 6, 1998, issued Mar. 1, 2005 (Ex. 1005, “Hudis”);

U.S. Patent 6,493,719 B1 to Booth et al., filed July 26, 1999, issued Dec. 10, 2002 (Ex. 1006, “Booth”); and

U.S. Patent No. 6,546,415 B1 to Park, filed May 14, 1999, issued Apr. 8, 2003 (Ex. 1007, “Park”).

E. The Asserted Grounds

Petitioner challenges claims 1–35 of the ’586 patent on the following grounds (Pet. 15):

Reference(s)	Basis	Claim(s) Challenged
Hudis	35 U.S.C. § 103(a)	1, 2, 4–9, 11–13, 21, 22, 24–29, and 31–35
Hudis and Park	35 U.S.C. § 103(a)	3, 10, 23, and 30
Hudis and Booth	35 U.S.C. § 103(a)	14, 15, and 17–20
Hudis, Booth, and Park	35 U.S.C. § 103(a)	16

F. Level of Ordinary Skill in the Art

Citing testimony from Dr. Fox, Petitioner contends a person having ordinary skill in the art would have had “at least a Bachelor’s degree in computer engineering or computer science (or an overlapping field) and 2-3 years of experience in the field of enterprise management.” Pet. 4–5 (citing Ex. 1003 ¶ 56). Petitioner further contends such a person would have had “experience[] in object-oriented methods, and in systems designed to using object-oriented principles.” *Id.* Patent Owner does not address the level of ordinary skill in the art.

For purposes of this Decision, we apply Petitioner’s definition without the qualifier “at least,” which introduces ambiguity. Based on the record

presented, we are satisfied that this definition comports with the level of skill necessary to understand and implement the teachings of the '586 patent and the asserted prior art.

G. Claim Interpretation

In an *inter partes* review, we construe claims in an unexpired patent by applying the broadest reasonable interpretation in light of the specification. 37 C.F.R. § 42.100(b); see *Cuozzo Speed Techs., LLC v. Lee*, 136 S. Ct. 2131, 2144–46 (2016). Under the broadest reasonable interpretation standard, and absent any special definitions, claim terms are given their ordinary and customary meaning, as would be understood by one of ordinary skill in the art in the context of the entire disclosure. See *In re Translogic Tech. Inc.*, 504 F.3d 1249, 1257 (Fed. Cir. 2007). Any special definitions for claim terms or phrases must be set forth “with reasonable clarity, deliberateness, and precision.” *In re Paulsen*, 30 F.3d 1475, 1480 (Fed. Cir. 1994).

1. “*hierarchical namespace*” / “*namespace . . . comprises a logical arrangement of . . . objects stored hierarchically*”

Each of independent claims 1, 14, 21, and 28 recites a “hierarchical namespace.” Independent claim 8 recites a “namespace” that “comprises a logical arrangement of . . . objects stored hierarchically.” We apply the following analysis to these similar recitations in the independent claims.

Petitioner contends that “hierarchical namespace” should be construed to mean “a logical arrangement of objects in a hierarchical, uniquely addressable system of object names.” Pet. 9. Petitioner bases its construction on the Board’s background statement about the '586 patent (but

not an express construction) in the ServiceNow IPR. *Id.* (citing Ex. 1011, 3). Petitioner cites several statements in the '586 patent specification as support for its construction. First, Petitioner contends a namespace “generally refers to a set of names in which all names are unique” and “*may* refer to a memory or plurality of memories coupled to one another, whose contents are uniquely addressable.” *Id.* (quoting Ex. 1001, 1:52–56, 11:22–27) (emphasis added by Petitioner). Second, Petitioner contends that “uniquely addressable” refers to “the property that items in a namespace have unique names such that any item in the namespace has a name different from all other names in the namespace.” *Id.* at 9–10 (citing Ex. 1001, 1:57–60, 11:27–30). Third, Petitioner contends that a “hierarchical namespace” is “a logical arrangement of objects, stored hierarchically.” *Id.* at 10 (citing Ex. 1001, 3:61–63, 27:14–16). Petitioner also asserts that “the '586 patent’s explanation that a namespace ‘may refer to a memory or plurality of memories . . .’ is merely exemplary.” *Id.* (citing Ex. 1003 ¶¶ 76, 80–82).

Patent Owner argues Petitioner’s proposed construction is incomplete because it omits a requirement that all names in the namespace are unique. Prelim. Resp. 22. According to Patent Owner, this requirement comes directly from the '586 patent specification:

As used herein, a “namespace” may refer to a memory, or a plurality of memories which are coupled to one another, whose contents are uniquely addressable. “Uniquely addressable” refers to the property that items in a namespace have unique names such that any item in the namespace has a name different from the names of all other items in the namespace.

Id. at 23–24 (quoting Ex. 1001, 1:50–60). Thus, Patent Owner proposes that we construe “hierarchical namespace” as “a memory or a plurality of memories coupled to one another containing a hierarchical set of objects

such that any object in the set has a different name from all other objects in the set.” *Id.* at 23. Patent Owner notes that this is the same construction adopted by the district court in the 2014 Texas litigation. *Id.* at 23–25 (quoting Ex. 1008, 32).

The parties agree that a “hierarchical namespace” is an arrangement of objects and that the objects are arranged hierarchically. They also agree that the “uniquely addressable” aspect should be included in the construction, but they differ on the specificity with which this concept should be described. We agree with Patent Owner that the construction must include the specification’s express definition of “uniquely addressable.” *See* Ex. 1001, 1:52–60 (“The term ‘namespace’ generally refers to a set of names in which all names are unique. . . . ‘Uniquely addressable’ refers to the property that items in a namespace have unique names such that any item in the namespace has a name different from the names of all other items in the namespace.”).

The parties also dispute whether a namespace must include “a memory or plurality of memories.” Patent Owner includes memories in its construction based on the specification, but Petitioner contends memories are merely an exemplary embodiment. We agree with Petitioner, because the references to memories in the specification are preceded by the permissive verbs “*may* refer” and “*may* be thought of.” *See* Ex. 1001, 1:54–55 (“[A] ‘namespace’ *may* refer to a memory, or a plurality of memories” (emphasis added)), 2:17–20 (“[A] namespace *may* be thought of as a plurality of distinct physical memories which are organized as a single, and possibly distributed, logical memory.” (emphasis added)). The specification also states that a namespace need not have the physical

form of a memory. *See id.* at 2:11–12 (“A namespace is typically a logical organization and not a physical one.”). Accordingly, we do not include a reference to memories in the construction. Instead, we agree with Petitioner that including a “logical arrangement of objects” in our construction better reflects the broadest reasonable interpretation of “namespace.”

For purposes of this Decision, we determine that the broadest reasonable interpretation of “hierarchical namespace” in claims 1, 14, 21, and 28, and “namespace” that “comprises a logical arrangement of . . . objects stored hierarchically” in claim 8, is “a logical arrangement containing a hierarchical set of objects such that any object in the set has a different name from all other objects in the set.”

2. *Other Terms*

Based on the current record, we determine that no other terms require explicit construction. *See Vivid Techs., Inc. v. Am. Sci. & Eng’g, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999) (“[O]nly those terms need be construed that are in controversy, and only to the extent necessary to resolve the controversy.”).

II. ANALYSIS

We now consider Petitioner’s asserted grounds and Patent Owner’s arguments in the Preliminary Response to determine whether Petitioner has met the “reasonable likelihood” standard for institution under 35 U.S.C. § 314(a).

A. *Obviousness Ground Based on Hudis*

Petitioner contends that claims 1, 2, 4–9, 11–13, 21, 22, 24–29, and 31–35 would have been obvious over Hudis. Pet. 16–54. Patent Owner disputes Petitioner’s contention. Prelim. Resp. 35–42.

1. *Hudis*

Hudis is a U.S. patent directed to “an improved system and method for providing management information to application programs.” Ex. 1005, 1:10–14. Figure 2 of Hudis is reproduced below.

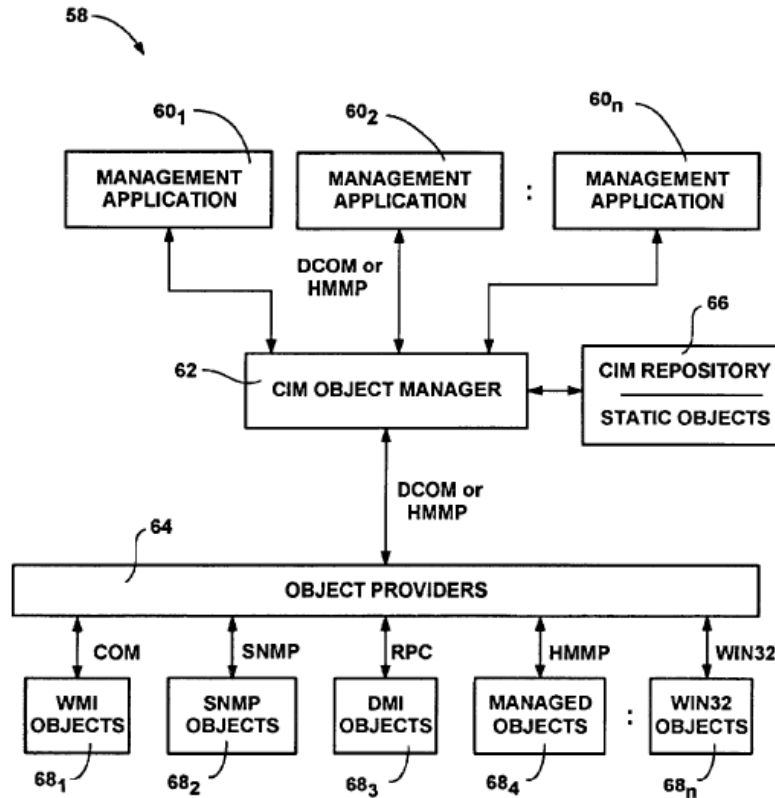


FIG. 2

Figure 2 depicts the architecture of the Common Information Model (CIM) for working with management information. *Id.* at 2:24–26. CIM is “a data representation formalism (schema) for managed objects.” *Id.* at 4:19–21. In

CIM installation 58, management applications 60₁–60_n are clients that communicate management information requests with a server known as CIM Object Manager (CIMOM) 62. *Id.* at 4:41–46. CIMOM 62 passes client requests to appropriate servers known as object providers 64. *Id.* at 5:1–4. Object providers 64 gather data from devices 68₁–68_n and return the data to CIMOM 62. *Id.* at 5:35–39.

“CIM uses an object-oriented paradigm, wherein manageable objects are modeled using the concepts of classes and instances.” *Id.* at 6:20–22. “Classes are models or templates for manageable objects, while instances are occurrences of classes” *Id.* at 6:34–36. Classes can be derived from a superclass, meaning that “[a] derived class implicitly inherits or contains the entire contents of the superclass.” *Id.* at 7:24–26. As used in Hudis, a “[n]amespace” is “a fundamental grouping unit for sets of classes and instances.” *Id.* at 9:31–32.

“[I]nstance identification is done through the use of keys.” *Id.* at 8:38–39. “[C]lasses inherit the key of any superclass, if there is one, and only if there is not a key value in the superclass chain is a class permitted to introduce its own key.” *Id.* at 9:9–12. “[T]he instances of a class are private to a particular namespace, i.e., instances with the same key may exist in another namespace, but two instances of a class with matching keys are considered to be the same instance within a namespace.” *Id.* at 9:33–37.

Figure 8 of Hudis is reproduced below.

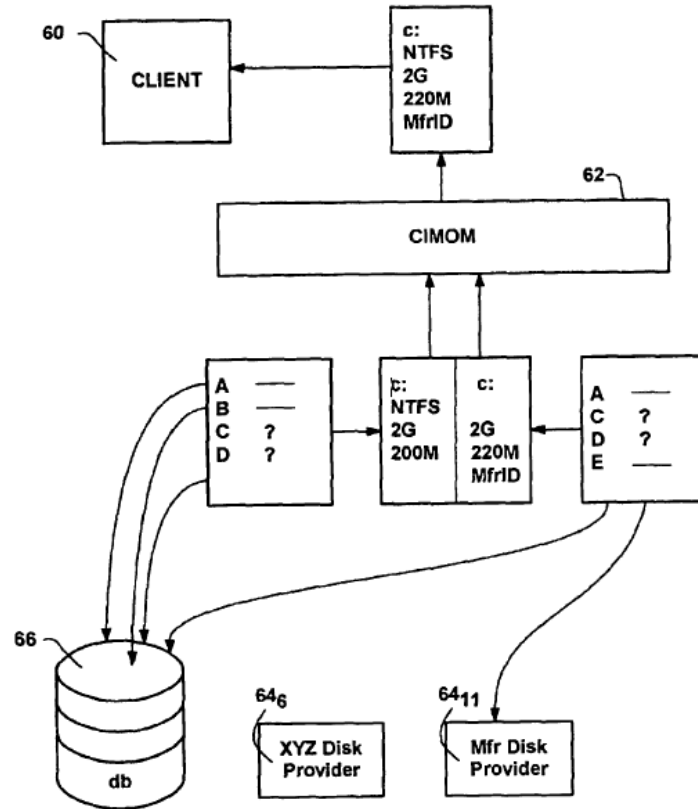


FIG. 8

Figure 8 depicts an example of how CIMOM 62 aggregates properties located for a class with inherited classes. *Id.* at 2:40–42. CIMOM 62 keeps the deepest of any common properties in the inheritance chain. *Id.* at 12:41–43. As shown in Figure 8, CIMOM 62 aggregates properties A, B, C, and D of the Disk instance (having values c:, NTFS, 2G, and 200M, respectively, and shown on the left side of Figure 8) with properties A, C, D and E of the Mfr instance (having values c:, 2G, 220M, and MfrID, respectively, and shown on the right side of Figure 8). *Id.* at 12:43–48. The joined result set (shown at the top of Figure 8) includes properties A through E, with any common values kept from the Mfr instance because that instance is deeper in the inheritance chain. *Id.* at 12:48–51.

2. *Claim 1*

A 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. *See 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 skill in the art; and (4) where in evidence, so-called secondary considerations. *See Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966). We also recognize that prior art references must be “considered together with the knowledge of one of ordinary skill in the pertinent art.” *Paulsen*, 30 F.3d at 1480 (citing *In re Samour*, 571 F.2d 559, 562 (CCPA 1978)). We analyze Petitioner’s obviousness grounds with the principles identified above in mind.

In its obviousness analysis for claim 1, Petitioner maps the recited “enterprise compris[ing] one or more networked computer systems” to Hudis’s personal computer 20 and remote computer 49, which are connected by LAN 51 and WAN 52. Pet. 16 (citing Ex. 1003 ¶ 128; Ex. 1005, 3:1–3, 3:55–57, 3:65–67). Regarding management of an enterprise, Petitioner cites Hudis’s management applications 60₁–60_n, which Petitioner contends are capable of managing objects through communication with CIMOM 62. *Id.* at 17 (citing Ex. 1003 ¶ 130; Ex. 1005, 1:10–13, 4:26–29, 4:41–50).

For the limitation “providing a hierarchical namespace,” Petitioner cites Hudis’s teaching that a namespace is a “fundamental grouping unit for

a set of classes and instances” that can be “nested in a manner similar to a hierarchical file system.” *Id.* at 17–18 (citing Ex. 1005, 6:33–34, 9:31–32, 9:45–46). Petitioner notes that classes and instances are said to be “objects” in Hudis. *Id.* at 17 (citing Ex. 1005, 6:33–34, 9:31–32). Petitioner contends that an object is uniquely addressable in the namespace and “identified by a ‘key’ (such as the volume name ‘C’), with which it can be located via an object path, such as ‘//myhost/root/default:disk.volume=“C”.’” *Id.* at 18 (citing Ex. 1003 ¶ 136; Ex. 1005, 8:38–43, 9:23–24). Citing testimony from Dr. Fox, Petitioner contends “each object within the namespace has a different name from any other object in the namespace” because of “the key (the property ‘C’ in this example).” *Id.* (citing Ex. 1003 ¶ 136). Petitioner further cites Hudis’s teachings on creating namespaces. *Id.* at 18–19 (citing Ex. 1005, 6:4–7, 9:52–54).

For the limitation “adding a plurality of objects to the namespace,” Petitioner cites Hudis’s “Put Class protocol operation” that can be used to create a class definition within a targeted server. *Id.* at 21–22 (citing Ex. 1003 ¶¶ 147–150; Ex. 1005, 6:23–25, 10:60–63). Regarding the limitation “objects [that] relate to software and hardware of the one or more computer systems,” Petitioner cites Hudis’s teaching on classes and instances (i.e., objects) containing various properties/attributes related to, for example, a disk. *Id.* at 19–20 (citing Ex. 1005, 7:55–63, 12:43–48, Fig. 8). Petitioner contends these objects are self-contained because they each contain data needed to model the disk. *Id.* at 20 (citing Ex. 1003 ¶¶ 142–143).

Regarding the limitation “sharing the plurality of objects with a plurality of the one or more computer system,” Petitioner cites Hudis’s

teaching that “CIMOM 62, in response to a management query from management applications 60₁-60_n, ‘returns an appropriate response via uniformly rendered managed objects.’” *Id.* at 23 (quoting Ex. 1005, 4:21–25 and citing *id.* at 11:63–65, 13:14–16). Petitioner contends an ordinarily skilled artisan would have found “sharing” obvious in view of these teachings from Hudis. *Id.* at 23 (citing Ex. 1003 ¶¶ 151–153). Petitioner explains that CINOM 62 “communicat[es] with management applications 60₁-60_n using, e.g., DCOM or HMMP protocols, rather than an interprocess communication method used when management applications and providers are on the same host.” *Id.* at 23–24 (citing Ex. 1003 ¶ 153; Ex. 1005, 4:46–50, 5:8–11). As such, Petitioner contends that an ordinarily skilled artisan would have recognized “that CIMOM 62 is designed to respond to queries of each of the management applications, thus sharing ‘a plurality of objects.’” *Id.* at 24.

Regarding the limitation that one object is a prototype and one object is an instance, Petitioner contends the object “Disk” in Hudis’s Figure 8 is a prototype and the object “Mfr” in Figure 8 is an instance. *Id.* at 25 (citing Ex. 1003 ¶ 160; Ex. 1005, 12:29–30, Fig. 8). To help explain its contention, Petitioner provides an annotated version of Hudis’s Figure 8, which is reproduced below.¹

¹ Dr. Fox includes another annotated version of Hudis’s Figure 8 in paragraph 160 of his declaration. The version in Dr. Fox’s declaration includes further annotations and appears to build upon the version appearing in the Petition. See Ex. 1003 ¶ 160.

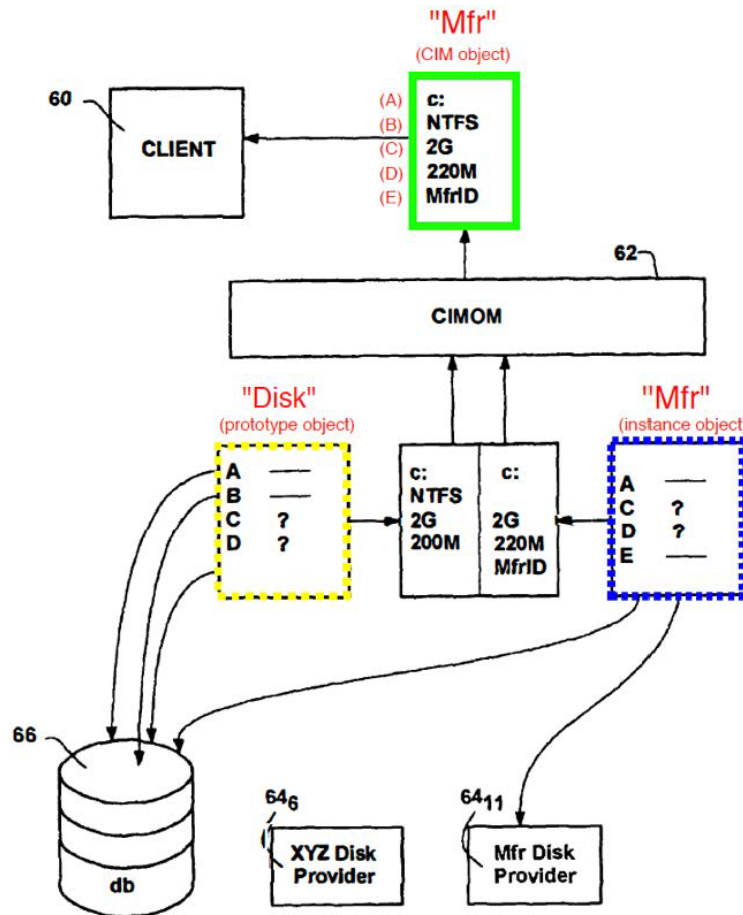


FIG. 8

Pet. 25. In this annotated version of Hudis’s Figure 8, Petitioner has indicated a “Disk” prototype object in yellow, a “Mfr” instance object in blue, and a “Mfr” CIM object in green. For the limitation “the instance dynamically inherit[ing] traits from the prototype,” Petitioner contends instance object “Mfr” dynamically inherits property “B” and the value of “B” (“NTFS”), both of which Petitioner calls “traits,” from prototype object “Disk.” *Id.* at 25–26 (citing Ex. 1003 ¶¶ 160, 165; Ex. 1005, 12:29–30, 12:41–51, Fig. 8). Regarding the “dynamic” aspect, Petitioner cites Hudis’s teaching that “most servers acting as CIMOMs will support class creation and update, . . . and instance creation and update” *Id.* at 27 (quoting

Ex. 1005, 10:60–63) (emphases added by Petitioner). More particularly, Petitioner cites the value of property “B” in the “Disk” object, which Petitioner notes is obtained from a database. *Id.* at 28 (citing Ex. 1005, 12:35–39, Fig. 8). Although Petitioner acknowledges that database information is “static” in Hudis, Petitioner contends an ordinarily skilled artisan would have known database values could change over time, such as from the “NTFS” file type to the “XFS” file type, both of which are disclosed in Hudis. *Id.* (citing Ex. 1003 ¶¶ 167, 169; Ex. 1005, 7:55–63, 12:35–39).

Regarding the limitation “the values of the traits inherited from the prototype change dynamically,” Petitioner cites the actions of Hudis’s CIMON 62. Pet. 29–30. In particular, Petitioner contends CIMOM 62 decomposes a received query for an object, retrieves the properties, and aggregates the properties, keeping the deepest of those common properties in the inheritance chain. *Id.* at 29 (citing Ex. 1005, 12:40–47). Petitioner further contends that objects resulting from the query are then sent to the requesting management application. *Id.* (citing Ex. 1005, 12:48–51). Petitioner argues that an ordinarily skilled artisan would have recognized that

CIMOM 62 performs this process *each time* a query for an object—including an object that inherits traits from another object—is received, and thus, to the extent the value of any trait in a prototype object is changed, it will be reflected in an instance that inherits from that prototype *at the time it is queried*.

Id. (citing Ex. 1003 ¶ 176). Petitioner then continues the example above related to property “B” in the “Disk” object changing from “NTFS” to “XFS.” Petitioner contends an ordinarily skilled artisan would have understood

from Hudis’s discussion of how CIMOM 62 retrieves and aggregates properties *at the time of the query* that the value of property “B” in the retrieved instance “Mfr” would change as or after the value of property “B” is changed in the prototype “Disk.” . . . This is because CIMOM 62 must resolve the chain of inheritance each time “Mfr” is queried, and “Mfr” will therefore always reflect the current value of property “B” in the prototype “Disk.”

Id. at 30 (citing Ex. 1003 ¶ 177; Ex. 1005, 11:12–22, 11:60–12:11, 12:40–51).

Patent Owner argues “Hudis does not disclose or suggest a ‘namespace’ because the classes and instances do not have to have a different name from all [ot]her classes and instances.” Prelim. Resp. 35. Petitioner relies on Hudis’s “key” for teaching unique naming, but Patent Owner argues Hudis’s teachings do not indicate unique naming. *Id.* at 35–36. Specifically, Patent Owner cites Hudis’s teaching that “instances *with the same key* may exist in another namespace, but *two* instances of a class with *matching keys* are considered to be the same instance within a namespace.” *Id.* at 36 (citing Ex. 1005, 9:34–37) (emphases added by Patent Owner). Patent Owner also cites a teaching from Hudis suggesting that classes and instances may not even have keys. *Id.* (citing Ex. 1005, 9:9–12) (“[C]lasses inherit the key of any superclass, *if there is one*, and only if there is not a key value in the superclass chain is a class permitted to introduce its own key.” (emphasis added)). Therefore, Patent Owner contends Petitioner “has not provided the evidentiary support or reasoned basis” demonstrating that Hudis teaches or suggests a “hierarchical namespace,” as recited in claim 1. *Id.*

We are persuaded by Patent Owner’s arguments. Petitioner relies on Hudis’s keys for teaching unique naming (*see* Pet. 18), but Hudis expressly

states that a namespace may have two instances with the same key. *See* Ex. 1005, 9:33–37. This does not comport with our construction of “hierarchical namespace” insofar as one object in the set (i.e., one instance) would not have a different name from all other objects in the set (i.e., the other instance with the same key). *See supra* § I.G.1. Patent Owner also establishes that certain superclass objects in Hudis may not even have keys, which Patent Owner likens to names. *See* Ex. 1005, 9:9–12. Without names, such objects cannot meet the unique naming requirements in our construction of “hierarchical namespace.” *See supra* § I.G.1. In addition, the same teaching cited by Patent Owner states that class objects can inherit keys from a superclass object, which undermines Petitioner’s assertion that a key is unique to an object. *See* Ex. 1005, 9:9–12. Petitioner has not explained why or how an ordinarily skilled artisan considering Hudis’s use of keys would have understood Hudis as teaching the unique naming of a “hierarchical namespace” consistent with our construction of that term.²

² Patent Owner suggests that another possible reading of Petitioner’s unpatentability contentions is that Petitioner is mapping the recited “hierarchical namespace” to “a hierarchy of namespaces” in Hudis. Prelim. Resp. 35. To the extent that is Petitioner’s contention, we agree with Patent Owner that “a hierarchy of namespaces (i.e., namespaces nested in a manner similar to a hierarchical file system) is not a hierarchical namespace because it says nothing about how the classes and instances in each namespace are logically arranged.” *Id.* More particularly, Petitioner has not explained how one object in the hierarchy of namespaces would have a different name from all other objects in the hierarchy of namespaces. *See supra* § I.G.1. This alternative theory would also fail based on the same deficiencies related to Hudis’s “keys” mentioned above, which would be endemic to each namespace among the “hierarchy of namespaces.”

We also have considered Dr. Fox’s testimony on Hudis’s teaching that “matching keys are considered to be the same instance within a namespace.” *See* Ex. 1003 ¶ 135 (quoting Ex. 1005, 9:33–37) (emphasis omitted). Dr. Fox testifies that this teaching resolves the problem of matching keys because “only a single instance object may have a given name (identified by the property designated as a ‘key value’) within a particular namespace.” *Id.* Yet merely considering two objects to be the same does not demonstrate that the two objects have different names. We do not credit Dr. Fox’s testimony in this regard because it is contradicted by Hudis. Dr. Fox also cites an example from Hudis related to “root/default” (*id.* (citing Ex. 1005, 8:60–64, 9:22–30)), but the example does not address the existence of two objects having the same name in Hudis. The fact that certain objects in Dr. Fox’s “root/default” example may have unique names does not indicate that all objects in the set have different names from one another, as required by our construction. *See supra* § I.G.1.

On this record, Petitioner has not established that Hudis teaches or suggests a “hierarchical namespace” as recited in claim 1. Accordingly, Petitioner has not established a reasonable likelihood that it would prevail in showing that the subject matter of claim 1 would have been obvious over Hudis.

3. *Claims 2, 4–9, 11–13, 21, 22, 24–29, and 31–35*

Petitioner’s obviousness analysis for claims 2, 4–9, 11–13, 21, 22, 24–29, and 31–35 relies on the same flawed analysis of “hierarchical

namespace” discussed above with respect to claim 1.³ See Pet. 30–54. Accordingly, we determine that Petitioner has not established a reasonable likelihood that it would prevail in showing that the subject matter of claims 2, 4–9, 11–13, 21, 22, 24–29, and 31–35 would have been obvious over Hudis.

B. Obviousness Ground Based on Hudis and Park

Petitioner contends the subject matter of claims 3, 10, 23, and 30 would have been obvious over Hudis and Park. Pet. 55–58. Patent Owner disputes Petitioner’s contention. Prelim. Resp. 42–43. Petitioner’s obviousness analysis for claims 3, 10, 23, and 30 relies on the same flawed analysis of “hierarchical namespace” discussed above with respect to the Hudis obviousness ground. See Pet. 55–58. Accordingly, we determine that Petitioner has not established a reasonable likelihood that it would prevail in showing that the subject matter of claims 3, 10, 23, and 30 would have been obvious over the combination of Hudis and Park.

C. Obviousness Ground Based on Hudis and Booth

Petitioner contends the subject matter of claims 14, 15, and 17–20 would have been obvious over Hudis and Booth. Pet. 58–65. Patent Owner disputes Petitioner’s contention. Prelim. Resp. 43. Like claim 1, claim 14 recites a “hierarchical namespace,” and Petitioner’s obviousness analysis for

³ We observe that Petitioner applies the same analysis of “hierarchical namespace” from claim 1 to the recitation in claim 8 that “the namespace comprises a logical arrangement of the objects stored hierarchically” and the recitations of a “hierarchical namespace” in claims 21 and 28. See Pet. 37, 43, 51–52.

claims 14, 15, and 17–20 relies on the same flawed analysis of “hierarchical namespace” discussed above with respect to the Hudis obviousness ground. *See* Pet. 58–65. Accordingly, we determine that Petitioner has not established a reasonable likelihood that it would prevail in showing that the subject matter of claims 14, 15, and 17–20 would have been obvious over the combination of Hudis and Booth.

D. Obviousness Ground Based on Hudis, Booth, and Park

Petitioner contends the subject matter of claim 16 would have been obvious over Hudis, Booth, and Park. Pet. 66–67. Patent Owner disputes Petitioner’s contention. Prelim. Resp. 43. Petitioner’s obviousness analysis for claim 16 relies on the same flawed analysis of “hierarchical namespace” discussed above with respect to the Hudis obviousness ground. *See* Pet. 66–67. Accordingly, we determine that Petitioner has not established a reasonable likelihood that it would prevail in showing that the subject matter of claim 16 would have been obvious over the combination of Hudis, Booth, and Park.

III. CONCLUSION

Petitioner has not demonstrated a reasonable likelihood of prevailing with respect to at least one claim of the ’586 patent challenged in the Petition. Therefore, we do not institute an *inter partes* review.

IV. ORDER

Accordingly, it is:

ORDERED that the Petition is *denied* as to all challenged claims of the '586 patent.

PETITIONER:

Eugene Paige
epaige@keker.com

PATENT OWNER:

Robert Auchter
rauchter@mckoolsmith.com
Brett Cooper
bcooper@mckoolsmith.com
Kevin Schubert
kschubert@mckoolsmith.com
Christopher P. McNett
cmcnett@mckoolsmith.com