NOTE: This disposition is nonprecedential.

United States Court of Appeals for the Federal Circuit

DATATERN, INC.,
Plaintiff-Appellant,

v.

EPICOR SOFTWARE CORPORATION,
Defendant-Appellee,

AND

INFORMATICA CORPORATION,
Defendant-Appellee,

AND

CARL WARREN & CO., INC.,
Defendant-Appellee,

AND

LANCET SOFTWARE DEVELOPMENT, INC.,
Defendant-Appellee,

AND

TERADATA CORPORATION,
Defendant-Appellee,

AND

PREMIER HEALTHCARE SOLUTIONS, INC.
(formerly known as Premier, Inc.)
Defendant-Appellee,

AND

MICROSTRATEGY, INC.,
Defendant-Appellee,

AND

AIRLINES REPORTING CORPORATION,
Defendant-Appellee,

AND

BLAZENT, INC.,
Defendant,

AND

MAGIC SOFTWARE ENTERTAINMENT, INC.,
AND MAGIC SOFTWARE ENTERTAINMENT, LTD,
Defendants.

2013-1251, -1252

Appeals from the United States District Court for the District of Massachusetts in Nos. 11-CV-11970 and 11-CV-12220, Judge F. Dennis Saylor, IV.

Decided: December 19, 2014

ERIK PAUL BELT, McCarter & English, LLP, of Boston, Massachusetts, argued for plaintiff-appellant. With him on the brief was LEE CARL BROMBERG.

ADAM J. KESSEL, Fish & Richardson P.C., of Boston, Massachusetts, argued for all defendant-appellees. On the brief for defendant-appellee Epicor Software Corporation was STEPHEN R. BUCKINGHAM, Lowenstien Sandler LLP, of Roseland, New Jersey. On the brief for defendant-appellee Informatica Corporation were J. DAVID HADDEN, DARREN E. DONNELLY, RYAN TYZ, and PHILLIP J. HAACK, Fenwick & West LLP, of Mountain View, California. On the brief for defendant-appellee Carl Warren & Co., Inc. was BRIAN P. VOKE, Campbell Campbell Edwards & Conroy, of Boston, Massachusetts. On the brief for defendant-appellee Lancet Software Development, Inc. were MICHAEL C. MCCARTHY and KEIKO L. SUGISAKA, Maslon Edelman Borman & Brand, LLP, of Minneapolis, Minnesota. On the brief for defendant-appellee Teradata Corporation was CRAIG R. SMITH, Lando & Anastasi LLP, of Cambridge, Massachusetts. On the brief for defendant-appellee Premier Healthcare Solutions, Inc., was MATTHEW E. LENO, McDermott Will & Emery LLP, of Boston, Massachusetts. On the brief for defendant-appellee Microstrategy Inc., was BENJAMIN K. THOMPSON, Fish & Richardson P.C., of Atlanta, Georgia. On the brief for defendant-appellee Airlines Reporting Corporation was DOMINIC MASSA, Wilmer Cutler Pickering Hale & Dorr, of Boston, Massachusetts.

Before LOURIE, MOORE, and CHEN, *Circuit Judges.*

MOORE, *Circuit Judge*.

DataTern appeals from the district court's entry of summary judgment that defendants do not infringe the

asserted claims of U.S. Patent No. 6,101,502. Because the claim construction is incorrect, we *vacate* and *remand*.

BACKGROUND

In the consolidated cases underlying this appeal, DataTern sued MicroStrategy and several of its customers (collectively, MicroStrategy) for infringing various claims of the '502 patent. At the same time, DataTern was involved in a declaratory judgment action involving the '502 patent in the United States District Court for the Southern District of New York. The New York court construed certain terms of the '502 patent, including the only term at issue on appeal in this case—"to create at least one interface object." *Microsoft Corp. v. DataTern, Inc.*, No. 11-cv-2365, 2012 WL 3682915, at *7–8 (S.D.N.Y. Aug. 24, 2012) (*New York Markman Order*). It construed this term to mean, "*to generate code for at least one class and instantiate an object from that class*, where the object is not part of or generated by the object oriented application and is used to access the database." *Id.* (emphasis added). In the present case, DataTern conceded that, if the district court in this case were to adopt the New York court's construction of "to create at least one interface object," then defendants do not infringe because they do not "generate code for at least one class and instantiate an object from that class." The district court granted summary judgment of noninfringement based solely on this concession. *DataTern, Inc. v. MicroStrategy, Inc.*, No. 11-11970-FDS (D. Mass. Feb. 7, 2013), ECF No. 108 (*Summary Judgment Order*). DataTern appeals. We have jurisdiction under 28 U.S.C. § 1295(a)(1).

DISCUSSION

The '502 patent is directed to interfacing an object oriented software application to access data stored in a

relational database. '502 patent col. 1 ll. 22–24, 53–55. An object oriented application cannot easily interface with a relational database because of the structural differences between the objects in the application and the tables in the database. *Id.* col. 1 ll. 25–49. To solve this problem, the '502 patent discloses creating "interface objects" that act as intermediaries between the object oriented application and the relational database. *Id.* col. 2 ll. 34–38. The patent discloses selecting an "object model," generating a map between the database schema and the object model, and creating the interface object using the map. *Id.* col. 2 ll. 28–34, 40–44. A "runtime engine" accesses data in the relational database using the interface object. *Id.* col. 2 ll. 34–38, Fig. 1. Claim 1 is representative:

> A method for interfacing an object oriented software application with a relational database, comprising the steps of:
>
> selecting an object model;
>
> generating a map of at least some relationships between schema in the database and the selected object model;
>
> employing the map *to create at least one interface object* associated with an object corresponding to a class associated with the object oriented software application; and
>
> utilizing a runtime engine which invokes said at least one interface object with the object oriented application to access data from the relational database.

Id. claim 1 (emphasis added).

The dispute in this case centers on the construction of "to create at least one interface object" as used in claim 1 and the '502 patent. The New York court construed this term to mean "to generate code for at least one class and instantiate an object from that class . . . ,"[1] New York Markman Order at *7–8, and the district court in this case adopted that construction and granted summary judgment of noninfringement, Summary Judgment Order at 2. The construction requires a two-step process: (1) generating code for a class; and (2) instantiating an object from that class. In support of this construction, the New York court noted that the phrase "employing the map" preceded "to create at least one interface object," meaning the map must be used to create the interface object. New York Markman Order at *7. It then reasoned that, in the embodiment of Figure 1 (below), the only way to generate interface objects (20) from the map (12) is through the code generator (18). Id. It recognized that Figure 1 also shows interface objects (20) connected to map (12) via runtime engine (24) without using code generator (18), but concluded that interface objects were not generated along this path in Figure 1 because "interface objects only

---

1 The construction of this term included a further limitation, "where the object is not part of or generated by the object oriented application and is used to access the database," based on a purported disclaimer in the prosecution history of the '502 patent. New York Markman Order at *7–8. The parties do not argue the correctness of this portion of the construction or otherwise explain how it would affect the infringement issue before us. Indeed, DataTern's concession of noninfringement was limited only to the first portion of the construction related to code generation for creating a class. J.A. 77–78. Thus, we express no opinion as to whether the further limitation included in the construction is correct.

come out of the code generator." *Id.* It also cited DataTern's expert's concession that he was not aware of any embodiments of the '502 patent "that do not require code generation." *Id.*

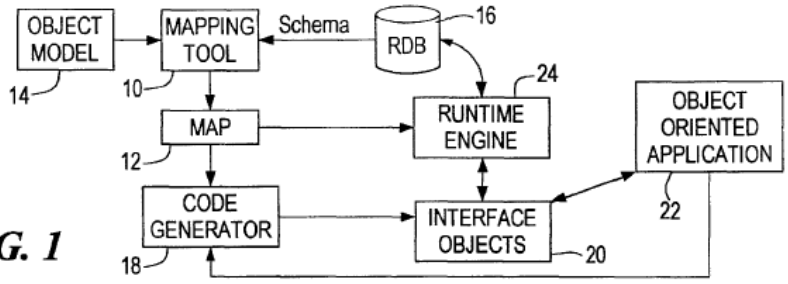

FIG. 1

We review claim construction *de novo*. *Lighting Ballast Control LLC v. Philips Elecs. N. Am. Corp.*, 744 F.3d 1272, 1276–77 (Fed. Cir. 2014) (en banc). We construe claim terms to have their ordinary and customary meaning, i.e., the meaning the term would have to a person of ordinary skill in the art in the context of the entire patent specification and its prosecution history. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1313 (Fed. Cir. 2005) (en banc).

We hold that the district court's construction of "to create at least one interface object" is incorrect. The plain language of the term and the context of the '502 patent both support the construction that "to create at least one interface object" is "to instantiate at least one interface object from a class." Claim construction begins with the plain language of the claims. The verb "to create" is readily understandable in common English and synonymous with "to make." The particular meaning of "to create" in the context of claim 1 of the '502 patent is informed by what is being created, in this case an interface object. As both parties agree, in object oriented applications, objects are created from classes by a process called "instantiation" and each object is said to be an

"instance" of its class. *See* Decl. of Neeraj Gupta at ¶ 9, *DataTern, Inc. v. MicroStrategy Inc.*, No. 11-12220-RGS (D. Mass. June 11, 2012), ECF No. 60 (referring to objects as "instances of data structures" and describing how these "object instances" perform the tasks required of the object oriented application); *id.* ¶ 20 ("Interface objects may be instantiated from pre-existing classes, or as in the preferred embodiment, from generated classes."); Appellees' Br. at 10–11 (discussing "'objects' that are created from 'classes'" via a "process . . . known as instantiation"). The specification and claims reinforce that objects are created by instantiation from classes. They refer repeatedly to various objects as an "object instance" or simply an "instance." '502 patent col. 2 ll. 55, col. 4 ll. 30–60, claims 22, 44; *see also id.* col. 6 ll. 20–40 (disclosing an object "aDslObject" being instantiated from the class "DPerson"); Appellees' Br. 24 ("[A] business object instantiates an object of the DPerson class, named aDslObject."); Reply Br. 13–15. In the New York case, DataTern stipulated to a construction of "class" as a "definition that specifies attributes and behavior of objects, and *from which objects can be instantiated*." J.A. 90. Thus, the plain meaning, specification, and record evidence support the conclusion that "to create at least one interface object" requires instantiating the interface object from a class.

To the extent, however, that the district court required generating code for a class as part of the claim step of creating "at least one interface object," it erred. To be clear, to instantiate an object from a class, you must have a class. That class, however, could be preexisting, or it could be generated as part of the overall object-creation process. The patentee chose to claim the instantiation of the object in the method step "to create at least one interface object." It did not, in this claim, limit the manner or timing in which the class comes into existence. Claim 1 recites a step of creating an interface object, but it does

not recite a preceding step of generating code for a class or, for that matter, generating code at all. Claim 1 is silent regarding code generation in connection with creating the interface object. In contrast, another independent claim recites using a "code generator" to create at least one interface object. '502 patent claim 10. We recognize that claims 1 and 10 differ in scope in other aspects. Nonetheless, viewing claim 1 in the context of claim 10 demonstrates that when the inventors of the '502 patent wanted to limit the claims to require code generation,[2] they did so explicitly. This suggests that claim 1 should not be so limited.

The phrase "employing the map" preceding the limitation at issue does not change this analysis. The parties debate the significance of this phrase, particularly in the context of the Figure 1 embodiment. MicroStrategy argues that "employing the map," together with Figure 1, requires generating code for a class because the only way to create interface objects using the map in Figure 1 is through code generator 18, which MicroStrategy asserts generates code for a class.[3] DataTern argues that Figure 1 discloses a second path where runtime engine 24 uses

---

[2] We note that even claim 10 requires a "code generator" without reciting what type of code is generated. Even claim 10 does not expressly require generating code *for a class*.

[3] We note that "it is improper to read limitations from a preferred embodiment described in the specification—even if it is the only embodiment—into the claims absent a clear indication in the intrinsic record that the patentee intended the claims to be so limited." *Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 913 (Fed. Cir. 2004).

the map 12 to create interface objects 20, without using code generator 18.

Figure 1 and the specification's sparse description of it do not support either party's contentions. The entirety of the description pertinent to this argument is:

> A code generator 18 is employed to examine the relationships that are defined in the map 12 and a model object oriented interface associated with an object oriented software application 22 to generate interface objects 20. The interface objects 20 are employed by the object oriented software application 22 to access the relational database 16 via a runtime engine 24, which also uses the map 12 to drive its processing.

'502 patent col. 2 ll. 32–38.

This portion of the specification explains that the code generator generates interface objects, and that the object oriented application, through the runtime engine, uses the map and interface objects to access the relational database. It does not disclose, as DataTern contends, that the runtime engine in Figure 1 generates objects using the map without using the code generator. It also does not disclose, as MicroStrategy contends, that generating code for a class is the only way to generate interface objects. In fact, neither Figure 1 nor the corresponding discussion in the specification even states that the code generator generates code *for a class*. Figure 1 does not support the district court's conclusion that the patentee intended code generation for a class to be a part of the claimed step of creating an interface object.

Figure 7, moreover, discloses an embodiment in which the runtime engine creates the interface object via a

process that does not include code generation for a class. MicroStrategy argues that Figure 7 is irrelevant to the construction of "to create at least one interface object" because Figure 7 only shows the operation of the runtime engine. Oral Argument at 20:09–20:28, *available at* http://oralarguments.cafc.uscourts.gov/default.aspx?fl=2013-1251.mp3. It contends that the runtime engine only performs the last step of claim 1, which expressly recites "utilizing a runtime engine," but not any of the other steps that do not include this requirement. We disagree. While MicroStrategy is correct that Figure 7 describes the operation of the runtime engine, '502 patent col. 2 ll. 23–24, claim 1 does not specify what performs the step of "employing the map to create at least one interface object." And it certainly does not preclude the runtime engine from doing so. The final two steps of claim 1 recite:

> employing the map to create at least one interface object associated with an object corresponding to a class associated with the object oriented software application; and
>
> utilizing a runtime engine which invokes said at least one interface object with the object oriented application to access data from the relational database.

Id. claim 1. The final step must be performed by the runtime engine, but claim 1—a method claim—is agnostic as to what performs the employing step. Again, this is in contrast to claim 10, which expressly requires the code generator to create the interface object and the runtime engine to invoke the interface object.

Figure 7 supports a construction that does not require generating code for a class because it describes an embod-

iment where the runtime engine employs the map to create an interface object from a preexisting class (i.e., without generating code for a class). At a high level, Figure 7 (below) discloses a multi-step process by which the runtime engine creates an object (steps 61–64) and uses the generated object to retrieve data (steps 65–68). '502 patent col. 6 ll. 31–64.

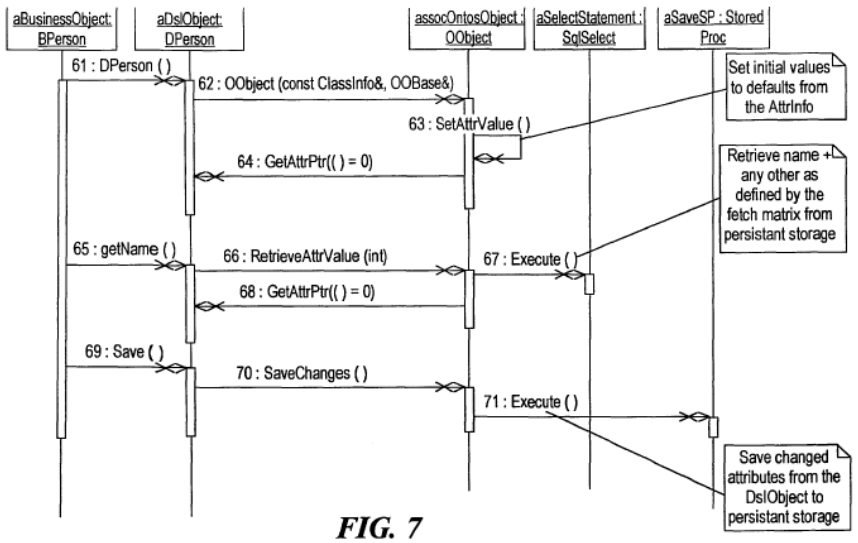

FIG. 7

As part of this process, Figure 7 discloses the instantiation of an interface object "aDslObject" from a class "DPerson." '502 patent Fig. 7 (steps 61–64), col. 6 ll. 31–44; Appellees' Br. at 24 ("[A] business object instantiates an object of the DPerson class, named aDslObject."); Reply Br. at 13–15. The specification describes Figure 7 as "the sequence of actions that take place when a business object *creates* a Dsl object," using the same verb—"create"—as claim 1. '502 patent col. 6 ll. 31–32 (emphasis added). As described in Figure 7, the interface object aDslObject is created by instantiating that object from the class DPerson. Generating DPerson itself is not part of the creation process. Generation of the underlying class is not what the specification refers to when it describes

"creat[ing]" an object. In fact, the specification clearly delineates between class generation and object creation or instantiation. For example, DPerson—the class from which the interface object aDslObject is instantiated—is described as "the generated . . . class." '502 patent col. 6 l. 36. Of course, common sense dictates that at some point, code must be generated for the DPerson class—it cannot miraculously come into existence without being generated. But Figure 7 describes generating code for a class and instantiating an object from that class as two different steps, and that the latter step creates the interface object.

Figure 7 also discloses "employing the map" to create the interface object. The steps used to generate the interface object (steps 61–64) use "AttrInfo" objects, which come from the map, to set certain object attributes to default levels. '502 patent col. 6 ll. 38–40, Figs. 4, 5 (showing AttrInfo as part of the map). Thus, Figure 7 shows employing the map to create the interface object.

CONCLUSION

For the foregoing reasons we construe "to create at least one interface object" as "to instantiate at least one interface object from a class."[4] Because the district court incorrectly construed the claim term upon which DataTern stipulated to noninfringement, we *vacate* the

---

[4] DataTern has raised a legitimate argument that MicroStrategy improperly relied on evidence of record in the New York case that is not of record in this case. We need not reach this issue, however, because the evidence in the New York case that MicroStrategy cites would not have changed the construction mandated by the plain language of claim 1, the '502 patent, and the evidence of record in this case.

grant of summary judgment of noninfringement and *remand* for proceedings consistent with this opinion.

VACATED AND REMANDED