Trials@uspto.gov
Tel: 571-272-7822

Paper 38
Entered: August 6, 2014

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

SAS INSTITUTE, INC.,
Petitioner,

v.

COMPLEMENTSOFT, LLC,
Patent Owner.

---

Case IPR2013-00226
Patent 7,110,936 B2

---

Before KEVIN F. TURNER, JUSTIN T. ARBES, and JENNIFER S. BISK, *Administrative Patent Judges.*

BISK, *Administrative Patent Judge*.

# FINAL WRITTEN DECISION

*35 U.S.C. § 318(a) and 37 C.F.R. § 42.73*

# I. INTRODUCTION

## *A. Background*

Petitioner, SAS Institute, Inc., filed a Petition (Paper 1, "Pet.") to institute an *inter partes* review of claims 1-16 ("the challenged claims") of U.S. Patent 7,110,936 B2 (Exhibit 1001, "the '936 patent"). 35 U.S.C. §§ 311-319. Patent Owner, ComplementSoft, LLC, filed a Preliminary Response. Paper 8. On August 12, 2013, we instituted trial (Paper 9, "Dec."), concluding that Petitioner had shown a reasonable likelihood of showing that claims 1 and 3-10 were unpatentable based on the following grounds:

| **References** [1] | **Claims Challenged** |
|---|---|
| Coad, Oracle Primer, and Oracle8 Primer | 1 |
| Antis and Coad | 1, 3, 5, 6, 8, and 10 |
| Antis, Coad, and Burkwald | 4 |
| Antis, Coad, and Eick | 7 |
| Antis, Coad, and Building Applications | 9 |

We have jurisdiction under 35 U.S.C. § 6(c). This final written decision is issued pursuant to 35 U.S.C. § 318(a) and 37 C.F.R. § 42.73.

[1] U.S. Patent No. 5,572,650 (Ex. 1005) ("Antis"); U.S. Patent No. 6,851,107 (Ex. 1006) ("Coad"); U.S. Patent No. 6,356,285 (Ex. 1007) ("Burkwald"); U.S. Patent No. 5,937,064 (Ex. 1008) ("Eick"); Microsoft Corporation, BUILDING APPLICATIONS WITH MICROSOFT ACCESS 97 (1996) (Ex. 1011) ("Building Applications"); Rajshekhar Sunderraman, ORACLE PROGRAMMING: A PRIMER (1999) (Ex. 1012) ("Oracle Primer"); and Rajshekhar Sunderraman, ORACLE8 PROGRAMMING: A PRIMER (2000) (Ex. 1013) ("Oracle8 Primer").

Petitioner has shown, by a preponderance of evidence, that claims 1, 3, and 5-10 are unpatentable. Petitioner has not met its burden to show that claim 4 is unpatentable.

Patent Owner's motion to amend claims is *denied.*

*B. Related Proceedings*

Patent Owner asserted the '936 patent against Petitioner in *ComplementSoft, LLC v. SAS Institute, Inc.*, No. 1:12-cv-07372 (N.D. Ill. Sept. 14, 2012). *See* Pet. 58; Paper 6 at 2. The related case is currently stayed pending this *inter partes* review. Transcript of Proceedings, *ComplementSoft*, No. 1:12-cv-07372, ECF No. 44 (granting stay), 54 (denying motion to lift stay).

*C. The '936 Patent*

The '936 patent describes a language independent software development tool having a graphical user interface, also referred to as an Integrated Development Environment or IDE. Ex. 1001, 1:15-19. In particular, the patent describes an IDE for exchanging, editing, debugging, visualizing, and developing software code for "data manipulation centric languages." *Id.* at 1:64-2:3.

The Summary of the Invention describes the IDE as including, among other features, a visualizer that generates a graphical representation of the program flow, data flow, or logic of the code. *Id.* at 2:34-49. In other words, the visualizer allows for displaying code in ways other than a typical text editor. A detailed description of a preferred embodiment uses a series of drawings, and corresponding text, to describe an exemplary IDE with a visualizer that can display source code using several different graphical formats. *Id.* at 3:24-26. For instance, Figure 9, reproduced below, depicts

"a program flow for a selected file, along with arrows that indicate the flow of data within the program flow." *Id.* at 3:49-51.

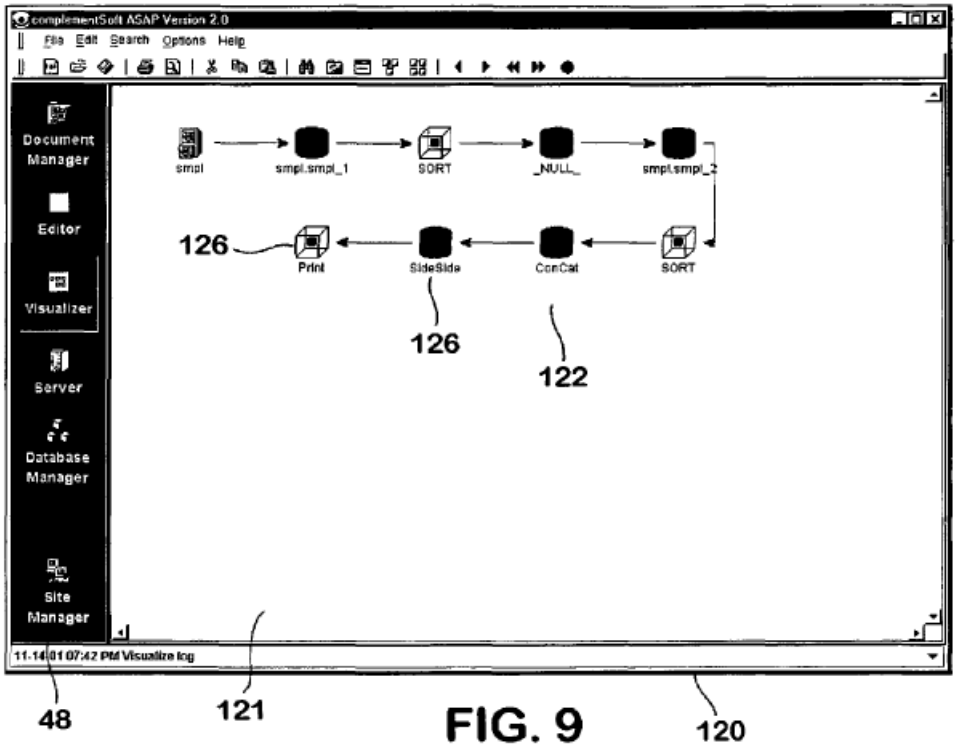


FIG. 9

Figure 9, above, shows visualizer 120 displaying source code. *Id.* Each program and data block of a code section is represented by an icon, program flow icon 126. *Id.* at 8:8-14. Program flow icons 126 are displayed in the order that they occur in the source code (*id.* at 15:56-59) and are connected by arrows that illustrate the flow of data (*id.* at 8:8-14).

Visualizer 120 also is shown in Figure 17, reproduced below, showing "a data flow" for the selected program. *Id.* at 4:12-13.

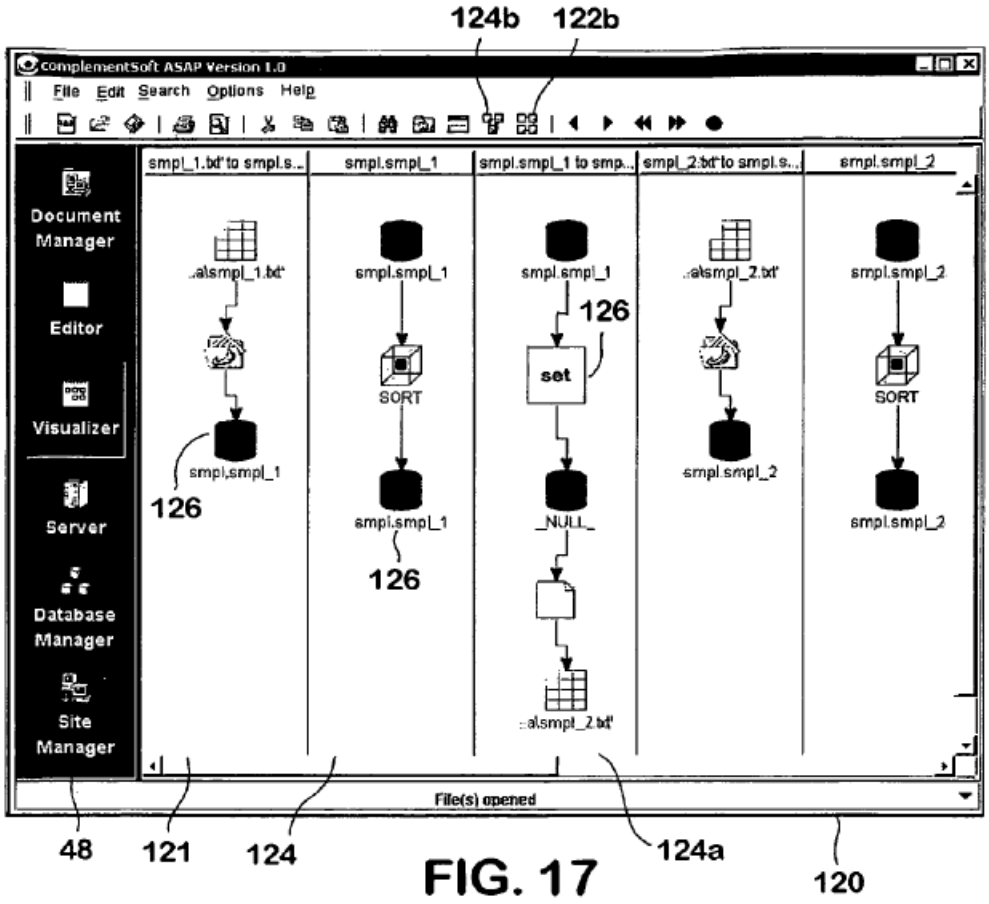


FIG. 17

Figure 17, above, shows visualizer 120 displaying individual processes and data blocks, represented by program flow icons 126, in separate columns. *Id.* at 16:6-12. The arrows that connect program flow icons 126 indicate the direction of the data flow. *Id.*

*D. Illustrative Claim*

Claim 1, reproduced below, is the '936 patent's only independent claim:

1. An integrated development environment, comprising:

a document manager for retrieving source code programmed using one of a plurality of types of data manipulation languages;

an editor for displaying the retrieved source code and providing a means for a user to edit the retrieved source code;

> a parser layer which detects the one of the plurality of types of data manipulation languages in which the retrieved source code is programmed and which activates rules and logic applicable to the detected one of the plurality of types of data manipulation languages; and
>
> a visualizer dynamically linked to the editor for displaying graphical representations of flows within the retrieved source code using the rules and logic applicable to the detected one of the plurality of types of data manipulation languages and activated by the parser,
>
> wherein the editor, parser layer and visualizer cooperate such that edits made to the source code using the editor are automatically reflected in the graphical representations of flows displayed by the visualizer and edits made to the graphical representations of flows in the visualizer are automatically reflected in the source code displayed by the editor.

## II. ANALYSIS

### *A. Claim Construction*

For purposes of the Decision to Institute we expressly construed the terms "data manipulation language" and "graphical representation of flows." Dec. 6-9. We construed (1) "data manipulation language" as "a programming language used to access data in a database, such as to retrieve, insert, delete, or modify data in the database," and (2) "graphical representation of flows" as "a diagram that depicts a map of the progression (or path) through the source code." *Id.*

In the post-institution briefs, the parties directly disagree regarding only the construction of the term "data manipulation language." Paper 16 ("PO Resp.") 10-11; Paper 24 ("Reply") 1-2. In analyzing the issues in this case, however, we have determined that many of the arguments purportedly directed to the proposed obviousness grounds are more accurately arguments regarding claim construction. Thus, to properly resolve the issues presented

in this proceeding, we construe several terms not addressed explicitly by either party, including "graphical representation of flows," "program flows," and "data flows." We construe all terms, whether or not expressly described below, using the broadest reasonable construction in light of the '936 patent specification. 37 C.F.R. § 42.100(b).

*1. Data Manipulation Language*

Much of this proceeding turns on the interpretation of the term "data manipulation language," recited by every challenged claim. In the Decision to Institute, we interpreted this term as "a programming language used to access data in a database, such as to retrieve, insert, delete, or modify data in the database." Dec. 6-8. This interpretation is consistent with dictionary definitions from the time period of the invention. *See* MICROSOFT COMPUTER DICTIONARY at 125 (4th ed. 1999) ("a language that is used to insert data in, update, and query a database"); Ex. 1040, 272 (THE AUTHORITATIVE DICTIONARY OF IEEE STANDARDS TERMS (7th ed. 2000)) ("A language used to retrieve, insert, delete, or modify the data in a database.").

*a. "retrieve"*

In its response brief, Patent Owner contends that the definition we adopted in the Decision to Institute is too broad. PO Resp. 10-11. Specifically, Patent Owner argues that a programming language that includes *only* the functionality for retrieving data would not have been considered a data manipulation language by a person of ordinary skill in the art. *Id.* Patent Owner's declarant, Ivan Zatkovich, testifies that an SQL [Structured Query Language] SELECT statement—which retrieves data from a database—"does not alter the data in any way and thus does not, by

itself, perform any manipulation of data." Ex. 2001 ¶ 19. Accordingly, Patent Owner proposes that our interpretation of the term "data manipulation language" be altered to either remove the word "retrieve" or "qualify the use of the term retrieve by stating that the retrieval must be followed by some *manipulation* procedure." PO Resp. 11. Petitioner disagrees, asserting that Patent Owner's proposed change would result in too narrow an interpretation because retrieval of data *does* constitute data manipulation. Reply 1-2.

We decline to make Patent Owner's proposed change. First, this change does not affect the substantive analysis in this case. In other words, our decision on patentability is the same whether or not we adopt Petitioner's proposed modification.

Second, we are not persuaded by Patent Owner's arguments that we should depart from the dictionary definitions, proffered by Petitioner's declarant, Dr. Nick Roussopoulos (Ex. 1015 ¶ 48), and upon which we based our preliminary construction. For instance, during the oral hearing, Patent Owner conceded that retrieving data from a database is a type of manipulation.

> JUDGE TURNER: But if I'm obtaining a smaller set [of data items from a database], isn't that manipulating or is it not? I understand I'm giving you a hypothetical and putting you on a spot.
>
> MR. HANFT: What I'm having trouble with is in that hypothetical you're talking about retrieving data from a dataset, but then you're trying to say, well, is that within the definition of a data manipulation language? It's just kind of slightly apples and oranges because the data manipulation language has to have certain characteristics and be capable of certain things.
>
> *Taking a dataset and reducing it down to a smaller set according to some characteristics is some type of manipulation,* but a

> data manipulation language has to be able to do more than just retrieve. It's got to do more than that.

Paper 36 ("Tr.") 38:7-18 (emphasis added). Fen Hiew, one of the named inventors of the '936 patent, agreed with this understanding. Ex. 1045, 47:15-18 (Q: "Would selection of data be a type of manipulation that's performed by a data manipulation system?" A: "Yes").

We are not persuaded otherwise by Mr. Zatkovich's testimony to the contrary. *See* Ex. 2001 ¶ 19. Mr. Zatkovich does not explain why retrieving data would not be considered manipulation of that data. Nor does he point to any objective evidence to support this conclusion. And Patent Owner does not point to persuasive language in the specification or other evidence that supports an interpretation of "data manipulation language" with the word "retrieve" removed or qualified as proposed.

Thus, we decline to alter our preliminary construction of data manipulation language by removing or adding a qualification to the word "retrieve."

*b. "directly"*

Although not couched as claim construction, Patent Owner argues that a data manipulation language must *directly* access data in a database. PO Resp. 33, 38-39. Patent Owner argues that because of this requirement, an object-oriented programming language cannot be a data manipulation language, even if it includes extensions, such as JDBC [Java Database Connectivity] or embedded SQL, for accessing a database. *Id*. In explaining this assertion, Mr. Zatkovich testifies that object-oriented programming languages facilitate the creation of programs that are a collection of interacting objects, as opposed to conventional programming languages, in

which a program is a list of tasks. Ex. 2001 ¶ 22. As part of this paradigm, according to Mr. Zatkovich, the object-oriented approach typically places data in an object, where the data are not directly accessible by the rest of the program, but instead are accessed solely through methods bundled with the object. *Id.* ¶ 23. Mr. Zatkovich concludes that "[o]ne skilled in the art would not consider C++ and Java [which are object-oriented programming languages] to be data manipulation languages since they do not directly interact or directly perform data manipulation within databases." *Id.* ¶ 24.

Patent Owner adds that simply adding database functionality in the form of JDBC or embedded SQL to an object-oriented language does not convert the language into a data manipulation language. PO Resp. 38-39. Mr. Zatkovich testifies that when SQL is embedded in Java, the Java program simply passes the SQL statement to the database system—the embedded SQL statement is "processed merely as a text string to be passed to the DataBase Management System." Ex. 2001 ¶ 47. According to Patent Owner, this shows that it is not actually Java or C++ code accessing data in a database, but instead the access is "being performed at and by the database itself." PO Resp. 39. Thus, Patent Owner concludes that Java and C++ are not data manipulation languages, even when augmented by JDBC or embedded SQL. *Id.*

Petitioner argues that a data manipulation language is a language that allows a program to simply access data in a database, without the requirement that the access be *direct.* Reply 9. According to Petitioner, Java and C++ access a database using embedded SQL and JDBC statements. *Id.* Petitioner relies on statements in the Oracle8 Primer to support this assertion. *Id.* at 9-10 (quoting Ex. 1013, 225 ("JDBC is an Application

Programming Interface (API) that enables database access in Java."), 93, 96, 226). In addition, Dr. Roussopoulos testifies that "[e]mbedding SQL allows each of [the Java and C++] programming languages to access data in a database." Ex. 1015 ¶ 114 (citing Ex. 1013, 93, 95, 103, 108, 118, 277, 280, 281, 294, and 301); *see also* Ex. 1015 ¶¶ 49-51. Dr. Roussopoulos also states that "Oracle8 Primer discloses that the object-oriented nature of Java is no bar to having data manipulation language operations." *Id.* ¶ 52 (quoting Ex. 1013, 281 (showing that the "select" statement is translated into pure Java code), 301 (showing the same for inserting data in a database), 294 (showing the same for creating database tables and rows)); *see also* Ex. 1015 ¶ 53 (showing the same for embedded C++).

We are persuaded by the testimony of Dr. Roussopoulos that a data manipulation language does not require *direct* access to data in a database. To the extent that Patent Owner argues that Java and C++ never actually retrieve or manipulate data in a database because the embedded functionality does not *directly* access the database, we credit Petitioner's evidence, particularly Exhibit 1013, which supports Dr. Roussopoulos's conclusion that embedded SQL accesses and manipulates data in a database. All of Patent Owner's evidence to the contrary hinges on the testimony of Mr. Zatkovich, which we do not find persuasive. Mr. Zatkovich's conclusion that access to a database from a data manipulation language must be direct is unsupported. Mr. Zatkovich simply asserts this to be the case, without providing credible support. *See* Ex. 2001 ¶ 47.

Patent Owner does not point to persuasive language in the specification or other evidence that supports an interpretation of data manipulation language restricted to *direct* access to a database. In fact, the

'936 patent discusses, in several places, SQL and Oracle® RDBMS as examples of languages to which the invention is targeted. Ex. 1001 1:20-25; 1:64-2:3; 8:22-26; 9:47-53; 10:5-7; 16:34-49; 17:17-19. And the '936 patent does not specify that these languages would be excluded if the database access functionality is indirect. Dr. Roussopoulos agrees, stating that the '936 patent's reference to these languages is consistent with embedding SQL in Java or C++. Ex. 1015 ¶ 54.

Thus, we decline to alter the interpretation of "data manipulation language" by adding a qualification that access to the database be direct.

*c. Conclusion*

For these reasons, we adopt the interpretation of "data manipulation language" used in the Decision to Institute—"a programming language used to access data in a database, such as to retrieve, insert, delete, or modify data in the database." We do not adopt any of the modifications to this interpretation urged by Patent Owner.

*2. Graphical Representation of Flows*

The term "graphical representation of flows" is recited by every challenged claim. In the Decision to Institute, we interpreted this term as "a diagram that depicts a map of the progression (or path) through the source code." Dec. 8-9. Patent Owner does not explicitly challenge this interpretation, but makes several arguments that amount to a narrowing of the interpretation of the term. We address these arguments here.

*a. "data flows" and "program flows"*

In the Decision to Institute, we explained that the '936 patent explicitly discusses two types of flow diagrams—"program flow diagrams" and "data flow diagrams." Dec. 8-9 (citing Ex. 1001, 2:38-42). The '936

patent describes a "program flow diagram" as being "comprised of program block icons and arrows to depict the code's program flow" and a "[d]ata flow diagram" as "comprised of icons depicting data processing steps and arrows to depict the flow of the data through the program." Ex. 1001, 2:38-42. We were not persuaded, however, that the claim term "graphical representation of flows" is restricted to these two types of flow diagrams. Dec. 8-9.

Patent Owner does not explicitly argue otherwise. Nevertheless, in its analysis, Patent Owner often limits the term to either program or data flow diagrams. These arguments are appropriate only for dependent claims 3 ("wherein the graphical representations of flows depict program flows") and 4 ("wherein the graphical representations of data flows are expandable and collapsible"). Patent Owner goes further and applies the narrower interpretation to all the claims. For example, Patent Owner argues that Coad does not show "graphical representations of flows" by asserting that the "communications shown in Fig. 14 are not directly related to program or data flow" and "none of the remaining diagrams of Coad show program or data flow." PO Resp. 31-32.

Patent Owner, however, presents no persuasive explanation or evidence to support such a narrow interpretation of the claim term "graphical representations of flows." As we pointed out in the Decision to Institute, Patent Owner has not directed us to language in the '936 patent that limits the term to only these two examples. Nothing in Patent Owner's response brief persuades us otherwise.

*b. "within the retrieved source code"*

Patent Owner argues that the term "graphical representations of flows" cannot properly be interpreted without taking into consideration the phrase that follows it in the claims—"within the retrieved source code." PO Resp. 32-33. According to Patent Owner, because the source code is further defined in the claims to be a "data manipulation language," the "claims require that the flows show source code steps actually performing data manipulation procedures." *Id.* at 35. Patent Owner explains that the "intent and purpose of the invention is to permit the graphical representation of the flow within languages that manipulate data." *Id.* at 36. Patent Owner also points to Figure 19 as showing that the graphical representations depict the program flow within the "*actual source code that is manipulating the data*." *Id.*

We agree with Patent Owner that the claim language clearly requires the graphical representations of flows to show flows within source code created with a data manipulation language. However, we are not persuaded that this requires the flows to show source code steps that are actually performing data manipulation. As described above, we have interpreted data manipulation language to be a programming language used to access data in a database, such as to retrieve, insert, delete, or modify data in the database. Although this interpretation requires that a data manipulation language allow a program to be written that manipulates data, the interpretation does not *limit* such a program to source code that manipulates data. In other words, source code written using a data manipulation language performs many types of actions, including, but not limited to, manipulating data.

For example, Figure 20 of the '936 patent shows a portion of the source code being displayed in the graphical representation. One of the function calls listed in the source code is "PRINT," which is represented in the graphical representation as an icon. It is unclear, however, whether a print function is "actual manipulation of data." *See* Tr. 65:19-23 (Q: "In the patent in figure 20(a) there is some code, and it has print data. Is that a data manipulation procedure? I don't know if printing is manipulating." A: "You're not going to like my answer, which is I'm not an expert in this field, so I can't answer that."). And Patent Owner has not directed us to evidence that sheds light on this question.

Moreover, as discussed above, Patent Owner asserts that retrieving data, as in an SQL SELECT statement, is not a manipulation of data. PO Resp. 10-11 (citing Ex. 2001 ¶ 19). Nevertheless, Patent Owner concedes that a program written with a data manipulation language may include functions used for retrieving data. *See, e.g.,* PO Resp. 11 ("The Board's interpretation should be modified to either remove the term 'retrieve' or to qualify the use of the term retrieve by stating that the retrieval must be followed by some *manipulation* procedure."). Thus, it is undisputed that a program written using a data manipulation language may contain portions of code that perform actions independent from the manipulation of data. And Patent Owner does not point to persuasive language in the '936 patent or other evidence supporting an interpretation that excludes those portions of the source code from the graphical representation. In other words, the claims require that the "received source code" is "programmed using one of a plurality of types of data manipulation languages," but nothing in the

claims requires that the "retrieved source code" contain functionality that actually manipulates data.

Figure 19 does not persuade us otherwise. Patent Owner has not pointed to any indication in the '936 patent that this figure is meant to limit the subject matter of the claims. To the contrary, the "Brief Description of Drawings" clearly identifies all the figures as depicting a preferred embodiment of the invention. Ex. 1001, 3:24-26 ("For a better understanding of the invention, reference may be had to a preferred embodiment shown in the following drawings."). And the "Detailed Description" concludes by clarifying that the specific embodiments are not limiting. *Id.* at 17:62-66 ("While specific embodiments of the present invention have been described in detail, it will be appreciated by those skilled in the art that various modifications and alternatives to those details could be developed in light of the overall teachings of the disclosure."); 18:13-17 ("Accordingly, the particular arrangement disclosed is meant to be illustrative only and not limiting as to the scope of the invention which is to be given the full breadth of the appended claims and any equivalents thereof.").

Thus, we decline to alter the interpretation of "graphical representations of flows" by adding a requirement that the flows show source code steps that are actually performing data manipulation.

*c. Actual Pathways*

Petitioner states that Patent Owner improperly suggests that graphical representations of flows are limited to the *actual* path through source code as opposed to including *all possible* pathways through the code. Reply 8-9. In particular, Petitioner points to Patent Owner's description of Coad, "[t]he

concept underlying Coad is limited to depicting all potential (as opposed to actual) scenarios within an object-oriented program." Reply 8 (citing PO Resp. 13). Patent Owner does not point to persuasive explanation or evidence supporting such a limitation. To the extent that Patent Owner is making this argument, we are not persuaded that the term should be so limited.

*d. Conclusion*

For these reasons, we adopt the interpretation of "graphical representations of flows" used for the Decision to Institute—"a diagram that depicts a map of the progression (or path) through the source code." We do not adopt any of the modifications to this interpretation urged by Patent Owner.

*3. Program Flows*

The term "program flows" is recited in dependent claim 3. In the Decision to Institute, we did not explicitly interpret this term as part of the claim construction section. *See* Dec. 5-10. In the analysis portion of our decision, however, we stated that "Figure 14 [of Coad] depicts the source code program's path of control from one step to another through the program—a program flow diagram." Dec. 14. Neither party directly challenges this statement.

Although the '936 patent does not explicitly define "program flows," it does define the term "program flow diagrams" as "comprised of program block icons and arrows to depict the code's program flow." Ex. 1001, 2:38-40. The specification then proceeds to use the terms "program flows" and "program flow diagrams" interchangeably. *See, e.g., id.* at 3:49-51 ("FIG. 9 is an exemplary screen shot depicting a program flow for a selected file,

along with arrows that indicate the flow of data within the program flow."); 16:3-5 ("By assigning meanings and attributes to tokens 144, the document view engine 200 allows the visualizer to create program flows 122 and data flows 124.").

Thus, we begin with the definition for "program flow diagrams" for our interpretation. Because it is not constructive for the definition of "program flows" to include the term "program flow," we adopt the following slightly modified version—"a graphical representation comprised of program block icons and arrows to depict the progression of control through source code."

*4. Data Flows*

The term "data flows" is recited by dependent claim 4. In the Decision to Institute, we did not explicitly interpret this term as part of the claim construction section. *See* Dec. 5-10. In the analysis portion of our decision, when discussing the data flow limitation, we stated that "[w]e are not persuaded that [the code view of Antis] is equivalent to a depiction of a map of the path of data through the executing source code." Dec. 19. Petitioner argues that the word "executing" in that statement is improper. Reply 4.

Although the '936 patent does not explicitly define "data flows," it does define the term "[d]ata flow diagrams" as "comprised of icons depicting data processing steps and arrows to depict the flow of the data through the program." Ex. 1001, 2:40-42. The specification then proceeds to use the terms "data flows" and "data flow diagrams" interchangeably. *See, e.g., id.* at 4:12-13 ("FIG. 17 is an exemplary screen shot depicting a data flow for a selected file."); 16:3-5 ("By assigning meanings and

attributes to tokens 144, the document view engine 200 allows the visualizer to create program flows 122 and data flows 124.").

Thus, we begin with the definition for "data flows diagrams," for our interpretation. Because it is not constructive to interpret the term "data flows" by using the phrase "flow of data," we adopt the following slightly modified version—"a graphical representation comprised of icons depicting data processing steps and arrows to depict the movement of data through source code."

*B. Overview of Coad*

Coad discloses a software development tool that allows a developer simultaneously to view and modify textual and graphical displays of source code regardless of the programming language in which the code is written. Ex. 1006, Abstract, 4:38-41. In the Background of the Invention, Coad describes conventional software development tools that allow the user to view Unified Modeling Language (UML)—a graphical representation or model using object-oriented design—and source code at the same time. *Id.* at 1:47–2:22.

*C. Overview of the Oracle Primers*

The Oracle Primers are books describing the Oracle database system. The Oracle8 Primer includes a chapter titled "Embedded SQL," which refers to adding embedded SQL to C++, thus allowing writing application programs in C++ that "interact (read and write) with the database." Ex. 1013, 93. Another chapter, titled "Oracle JDBC" describes JDBC, "an Application Programming Interface (API) that enables database access in Java" and "consists of a set of classes and interfaces written in Java that allow the programmer to send SQL statements to a database server for

execution and, in the case of an SQL query, to retrieve query results." Ex. 1013, 225.

*D. Alleged Obviousness over Coad and the Oracle Primers*

Petitioner asserts that claim 1 would have been obvious over Coad combined with Oracle Primer and Oracle8 Primer. Petitioner relies on Coad for every limitation except that Petitioner relies on the Oracle Primers for describing the use of SQL within Java and C++ and thus disclosing the data manipulation language limitation. Pet. 31-32 (citing Ex. 1015 ¶¶ 111-115). Petitioner points to Figures 11-17 of Coad as depicting aspects of the view for displaying graphical representations of flows in source code. Pet. 29-30.

In the Decision to Institute, we determined that Petitioner had shown a reasonable likelihood of prevailing on this proposed ground of unpatentability. Dec. 15. In particular, we determined that Petitioner had a reasonable likelihood of prevailing on its assertions that the combination of Coad and the Oracle Primers disclosed every limitation of claim 1. *Id.* at 12-14. We also found reasonable Petitioner's asserted rationale that a person of ordinary skill would have combined the teachings of Coad and the Oracle Primers in order to enhance the utility of the programming environment to include data manipulation. *Id.* at 14-15 (citing Pet. 25); *see* Ex. 1015 ¶ 115.

In its response brief, Patent Owner argues that the combination of Coad and the Oracle Primers fails to disclose the limitations "graphical representations of flows within the retrieved source code" where the source code is written in a "data manipulation language." PO Resp. 34-42. Patent Owner does not address any other limitations of claim 1 or the rationale to combine the references. *Id.*

*1. Data Manipulation Language*

Patent Owner argues that Coad combined with the Oracle Primers does not describe a data manipulation language because C++ and Java are object-oriented languages. PO Resp. 13, 36-39. According to Patent Owner, combining these languages with embedded SQL or JDBC, as disclosed by the Oracle Primers, does not solve the problem because the database access is not direct. *Id.*

As described above, our interpretation of the term "data manipulation language"—a programming language used to access data in a database, such as to retrieve, insert, delete, or modify data in the database—is broad enough to encompass object-oriented languages that do not directly access data in a database. We agree with Petitioner that when used with an embedded SQL or JDBC API, Java and C++ can be used to access data in a database and therefore qualify as data manipulation languages as we have construed that term. *See, e.g.*, Ex. 1015 ¶¶ 49-50 ("Java, C, C++, and other programming languages had functions and structures through the use of/embedding of SQL statements that allowed the programming language to access data, such as to retrieve, insert, delete, or modify data in a database.").

*2. Graphical Representations of Flows*

Patent Owner also argues that the combination of Coad and the Oracle Primers does not disclose the limitation "graphical representations of flows." PO Resp. 40-42. In particular, Patent Owner asserts that "[i]mplementing JDBC or embedded SQL within the environment of Coad would produce, at best, a graphical depiction of objects that show undefined external function calls, and would fail to show the program or data flow within the string quotes being passed to the database." PO Resp. 40.

We are not persuaded by this argument. First, this argument is based on an interpretation of the term "graphical representations of flows" that we rejected above. As explained, the interpretation we adopt is a diagram that depicts a map of the progression (or path) through the source code. This interpretation is broad enough to include, but is not limited to, program and data flows. We also reject Patent Owner's assertion that the graphical representation must show flow within source code steps that are actually performing data manipulation. Our interpretation of the term includes, but does not require, that the flows are shown within source code that is actually performing data manipulation.

Petitioner relies on Figures 11-17 (depicting UML diagrams) of Coad as disclosing graphical representations of flows. Pet. 28-30. We agree with Petitioner that at least Figures 14 and 17 of Coad disclose graphical representations of flows as we have construed that term. *See, e.g.*, Ex. 1015 ¶¶ 97-98; *see also* Ex. 1043, 88-89 (describing UML diagrams as showing flows).

Patent Owner concedes that Figure 14 shows communications between objects, which "would at best only show program flow in a purely object[-]oriented language—between objects." PO Resp. 32 (citing Ex. 2001 ¶ 36). Nonetheless, Patent Owner asserts that because object messages "cannot sensibly be asserted to constitute program or data flow," Figure 14 does not show flow through source code. *Id.* at 31-32. Because our interpretation of the term is broad enough to encompass flows that are not program or data flows, we conclude that Figure 14 discloses graphical representations of flows.

Similarly, Mr. Zatkovich agrees that Figure 17 shows a type of flow—"Figure 17 can depict a type of flow [], within an object, but only within state-based objects. So it's a very specialized type of flow in a very limited circumstance." Ex. 1044, 104:12-17. Because our interpretation of the term is broad enough to encompass flows that are not program or data flows, we conclude that Figure 17 discloses graphical representations of flows as well.

*3. Conclusion*

We conclude that a preponderance of the evidence demonstrates that claim 1 is unpatentable based on the combination of Coad and the Oracle Primers.

*E. Overview of Antis*

Antis relates to visually displaying structural characteristics of a large database in various graphical views for development purposes. Ex. 1005, Abstract. In particular, Antis describes a tool to display the characteristics of a database without semantic information such that explicit and implicit data structures can readily be observed to facilitate use, development, and maintenance of large databases. *Id.* 2:25-29.

*F. Alleged Obviousness over Antis and Coad*

As summarized in the table above, Petitioner asserts that claims 1 and 3-10 would have been obvious over Antis combined with Coad (claims 1, 3, 5, 6, 8, and 10) or Antis combined with Coad and one other reference (claims 4, 7, and 9). Pet. 41-52. In the Decision to Institute, we determined that Petitioner had shown a reasonable likelihood of prevailing on these proposed grounds of unpatentability. Dec. 18-19. In particular, we determined that Petitioner had a reasonable likelihood of prevailing on its assertions that the combination of asserted references discloses every

limitation of the challenged claims. *Id.* We also found reasonable Petitioner's asserted rationale that a person of ordinary skill would have combined the teachings of the references in order to allow for easier source code debugging and a more accurate code view display. *Id.* at 18 (citing Pet 18); *see also* Ex. 1015 ¶¶ 164-67.

*1. Independent Claim 1*

In its response brief, Patent Owner argues that the combination of Antis and Coad fails to disclose the limitations (recited in every challenged claim) "graphical representations of flows within the retrieved source code" where the source code is written in a "data manipulation language." PO Resp. 42-45. Patent Owner does not address any other limitations of claim 1 or Petitioner's asserted rationale to combine Antis and Coad. *Id.*

*a. Data Manipulation Language*

Patent Owner argues that Antis combined with Coad does not describe a data manipulation language because Antis instead describes the use of a data definition language. PO Resp. 42-45. A data definition language is designed specifically for describing the relationships between data in a database, such as defining data structures and schema. *Id.*

We agree with Patent Owner that a data definition language is different than a data manipulation language. *See* Ex. 1015 ¶ 48 (quoting The Authoritative Dictionary of IEEE Standards Terms at 100 (7th 2000) (defining data manipulation language followed by ("*Contrast:* data definition language"))). However, we do not agree that Antis's disclosure is limited to data definition languages.

Antis also discusses the use of RDBMS. *See, e.g.,* Ex. 1005, 3:30-35, 5:4-8; Ex. 1044, 138:19-139:12. The '936 patent expressly mentions

RDBMS several times, stating, for example, that "a need exists for a system and method for exchanging, editing, debugging, visualizing[,] and developing SAS®, SPSS®, SQL®, DB2 UDB®, Oracle RDBMS®[,] and other relational database management software." Ex. 1001, 1:66-2:3; *see also* 1:20-25; 8:26-30. Thus, the '936 patent contemplates the use of a database management system with the invention. *Id.* Mr. Hiew testifies that a data management system typically includes a data manipulation language to retrieve and manipulate the data from the storage management by the system. Ex. 1045, 47:19-48:12. Consistent with this definition, Antis shows some source code that retrieves data from a database—a data manipulation language. Ex. 1005, Fig. 12 ("/HOME/PYRCE/DATA/*EXTRACT*/V6.0/") (emphasis added); Ex. 2002, 101:4-102:19 (Dr. Roussopoulos testifying that although he is not familiar with the language the source code in Figure 12 is in, it shows a database query).

We are persuaded that Antis discloses a data manipulation language as we have construed that term.

*b. Graphical Representations of Flows*

Patent Owner also argues that the combination of Coad and Antis does not disclose the limitation "graphical representations of flows." PO Resp. 42-45. Patent Owner argues that Antis does not disclose graphical representations of flows because it only shows relations within a database, not any type of flow. *Id*. at 43-44. We are not persuaded by this argument. As explained above, we have found that Coad discloses graphical representations of flows, so it is irrelevant that Antis does not also show this particular limitation, given that the asserted ground is based on the combination of the two references.

*c. Conclusion*

We conclude that a preponderance of the evidence demonstrates that independent claim 1 is unpatentable based on the combination of Antis and Coad.

*2. Claims 5, 6, 8, and 10*

Claims 5, 6, and 8 depend directly from independent claim 1. Claim 10 depends from claim 8. Patent Owner does not separately argue the limitations added by these dependent claims. After considering all the papers filed in this proceeding, we are persuaded that dependent claims 5, 6, 8, and 10 are unpatentable based on the combination of Antis and Coad for the reasons argued by Petitioner.

*3. Claim 3*

Claim 3 depends directly from claim 1 and adds the limitation "wherein the graphical representations of flows depict program flows." Because claim 3 specifically limits the graphical representations to program flows, we revisit some of Patent Owner's arguments that we did not find persuasive when applied to the broader term. In other words, although we determined above that Coad discloses graphical representations of flows, we must now determine whether Coad shows *program flows* as we construe that term—a graphical representation comprised of program block icons and arrows to depict the progression of control through source code.

In our Decision to Institute, we stated that "Figure 14 [of Coad] depicts the source code program's path of control from one step to another through the program—a program flow diagram." Dec. 14. We based this determination on Coad's description that Figure 14, showing a sequence diagram, depicts "the time ordering of messages along the vertical axis"

representing "an interaction . . . to effect a desired operation or result." *Id.* (citing Ex. 1006, 17:1-15). Dr. Roussopoulos's testimony is consistent with this determination. Ex. 1015 ¶¶ 100, 101. In particular, Dr. Roussopoulos states that "Figure 14 depicts a program flow by showing a *particular sequence of operations*, where one operation follows another in time." *Id.* ¶ 100. He points to objective evidence supporting his conclusion that a person of ordinary skill would understand sequence diagrams to include program flows. *Id.* ¶ 101 (quoting Mehmet Aksit, *et. al*., *Use Cases in Object-Oriented Software Development*, AMIDST, Feb. 5, 1999, at 10-11); *see also id.* ¶ 102.

Patent Owner argues that Figure 14 does not disclose program flows because "the sequence diagrams show the *communications* that occur between objects, not the flow of program control between objects, nor the flow of data being manipulated by the objects." PO Resp. 31. Mr. Zatkovich testifies similarly, stating that "sequence diagrams show the communications between active objects" and there is "nothing in this type of model representation that is intended to show how data flows." Ex. 2001 ¶¶ 28-29. Thus, according to Mr. Zatkovich, "one skilled in the art reviewing Fig. 14 and the accompanying text would not conclude that this discloses program or data flow in Java or C++, nor program or data flow in a data manipulation language." *Id.* ¶ 30.

We are persuaded that a person of ordinary skill in the art would conclude that Figure 14 discloses program flows as we have construed that term. Patent Owner does not persuasively address the language of Coad itself—that sequence diagrams "emphasize the time ordering of messages along the vertical access" (Ex. 1006, 17:11-15), and thus depict the

progression of control through the source code of an object. Moreover, Patent Owner does not persuasively address the supporting evidence stating that "[t]he flow of control in use cases can be displayed in interaction diagrams, especially the sequence diagrams." Ex. 1015 ¶ 101 (emphasis omitted).

As between the conflicting evidence on this point, we credit Petitioner's evidence, particularly Dr. Roussopoulos's testimony, which is supported by objective evidence. All of Patent Owner's evidence, to the contrary, hinges on the testimony of Mr. Zatkovich, which we do not find persuasive on this point. Mr. Zatkovich's conclusion that a person of ordinary skill would not conclude that Figure 14 depicts program flows is unsupported. Mr. Zatkovich simply asserts this to be the case, without providing credible support. *See* Ex. 2001 ¶¶ 28-30.

We conclude that a preponderance of the evidence demonstrates that claim 3 is unpatentable based on the combination of Antis and Coad.

*4. Claim 4*

Petitioner asserts that claim 4 would have been obvious over Antis, Coad, and Burkwald. Pet. 52-53. Claim 4 depends from claim 1 and adds the limitation that "the graphical representations of data flows are expandable and collapsible." Petitioner relies on Burkwald—a patent directed to a "system for visually representing modification information about a[] characteristic-dependent information processing system"—as disclosing this limitation. *See id.* at 52 (citing Ex. 1007, 14:43 – 15:4); Ex. 1007, Title. Petitioner explains that a person of ordinary skill in the art would have had a reason to combine the three references because they are all related to software development tools that provide visual representations of

source code. Pet. 19. According to Petitioner, Burkwald's teaching of expanding and collapsing graphical representations of flows would have provided a developer with flexibility in the amount of detail shown in the view. *Id.* at 19-20. Patent Owner argues that the combination of Coad, Antis, and Burkwald does not disclose "data flows," but Patent Owner does not address any of the other limitations added by claim 4 or the rationale to combine the references. *See* PO Resp. 45-47.

Much like claim 3, claim 4 specifically limits the graphical representations, here to data flows. Thus, we must determine whether Coad shows *data flows* as we construe that term—a graphical representation comprised of icons depicting data processing steps and arrows to depict the movement of data through source code.

Petitioner argues that the UML sequence and collaboration diagrams of Coad show data flows. Pet. 43, 52-53; Reply 7. Consistent with this assertion, Dr. Roussopoulos explains that a person of ordinary skill in the art would understand Figure 14 to depict data flows. Ex. 1015 ¶¶ 97-98. According to Dr. Roussopoulos, "[t]he person of ordinary skill in the art would thus understand Figure 14 of Coad to disclose a graphical representation of data flow because it shows *which pieces of data* (*i.e.*, the data that is passed to the functions) *are accessed by which pieces of source code* (*i.e.*, the source code comprising the functions)." *Id.* ¶ 98.

We are not persuaded by Dr. Roussopoulos's unsupported conclusions on this point. For example, although his testimony addresses part of our interpretation of the term "data flows"—"arrows to depict the movement of data through source code," his testimony does not explain how Coad depicts "icons depicting data processing steps." Conversely, Dr. Roussopoulos

explicitly states that "[i]n Figure 14 of Coad, the horizontal dimension represents different objects" and "[i]n transitioning between the objects of the horizontal dimension, various functions are invoked." *Id.* Consistent with this testimony, Figure 14 appears to show "various functions" using arrows and objects using icons, but it is unclear that any icons represent "data processing steps" as required.

Dr. Roussopoulos also testifies that "a person of ordinary skill in the art would understand the statechart diagram of Figure 16 to disclose both data flows and program flows." *Id.* ¶ 103. This testimony relies on evidence that "a statechart diagram can be translated into a data flow diagram" leading a person of ordinary skill in the art to "understand that data flow must necessarily be depicted in a statechart diagram." *Id.* ¶ 104. Coad, however, explains that Figure 16 depicts "the sequences of states 1602 that an object or interaction goes through during its life." Ex. 1006, 17:16-20. Dr. Roussopoulos does not explain how this figure shows "icons depicting data processing steps." Dr. Roussopoulos's testimony that Figure 17, an activity diagram, depicts data flows suffers from the same problem. *Id.* ¶ 107.[2]

In its reply brief, Petitioner adds that the UML diagrams of Coad (Figures 14-17) depict the same types of data flows as shown in Figures 8-2 and 8-3 of the UML Manual (Ex. 1043). Reply 6. The figures depicted in the UML Manual, however, suffer from the same problem as we have

[2] This conclusion is consistent with our decision declining to institute an *inter partes* review of claim 2 because we were not persuaded that either Antis or Coad discloses the claimed "graphical representation" of a "data flow." Dec. 19.

identified for Figures 14, 16, and 17 of Coad—they do not appear to show "icons representing data processing steps."

We conclude that Petitioner has not shown by a preponderance of the evidence that dependent claim 4 would have been obvious to a person of ordinary skill in the art based on the combined disclosure of Coad, Antis, and Burkwald.

*5. Claims 7 and 9*

Claim 7 depends from claim 6 and adds the limitation that "the document manager comprises a security layer for managing secure connections with the one or more remote computers." Petitioner relies on Eick—a patent directed to a "system and method for interactive visualization, analysis and control of a dynamic database"—as disclosing this limitation. *See* Pet. 53 (citing Ex. 1008, 4:19-27); Ex. 1008, Title. Petitioner explains that a person of ordinary skill in the art would have had a reason to combine the three references because they are all related to visual representations of source code and data structures. Pet. 21. Moreover, according to Petitioner, Eick's teaching of a security layer for managing secure connections with remote computers would allow the systems of Antis and Coad to be distributed to one or more locations without a substantial security risk. *Id.* at 21-22.

Claim 9 depends from claim 8 and adds the limitation that "the template manager is adapted to automatically correct segments of the source code." Petitioner relies on Building Applications—a book including information about Microsoft Access 97 software—as disclosing this limitation. Pet. 54-55 (citing Ex. 1011, 52-54). Petitioner explains that a person of ordinary skill in the art would have had a reason to combine the

three references because they are all related to software development tools that provide visual representations of source code. Pet. 23. According to Petitioner, Building Applications's teaching of automatically correcting segments of source code determined to have errors would simplify the debugging of source code in Antis and Coad. *Id.* at 23-24.

Patent Owner does not separately argue these grounds, but instead states that "[t]hese claims are patentable for the same reasons as set forth . . . with respect to claim 1." PO Resp. 47. For the reasons discussed with respect to claim 1, and considering the record, we conclude that a preponderance of the evidence demonstrates that claim 7 is unpatentable based on the combination of Antis, Coad, and Eick, and claim 9 is unpatentable based on the combination of Antis, Coad, and Building Applications for the reasons argued by Petitioner.

*G. Patent Owner's Contingent Motion to Amend Claims*

Patent Owner filed a motion to enter proposed, amended claims 17-25, contingent on the Board determining that claims 1 and 3-10, respectively, are unpatentable. Paper 20 ("Mot. to Amend"). Patent Owner also filed a Second Contingent Motion to Amend solely addressing potential antecedent basis issues in the proposed substitute claims. Paper 28 ("Second Mot. to Amend"). Because we determine that claims 1, 3, and 5-10 are unpatentable, we consider the proposed substitute claims 17, 18, and 20-25. However, because we do not determine that claim 4 is unpatentable, we do not consider the proposed substitute for that claim—claim 19.

During an *inter partes* review, we enter proposed amended claims only upon a showing that the amended claims are patentable. *Idle Free Sys. v. Bergstrom, Inc.*, Case IPR2012-00027, slip op. at 33 (PTAB Jan. 7, 2014)

(Paper 66). This burden may not be met by merely showing that the proposed claims are distinguished over the prior art references applied to the original patent claims. Instead, because there is no examination of the proposed claims, the Patent Owner must show that the subject matter recited is not taught or suggested by the prior art for us to determine if they comply with 35 U.S.C. §§ 102 and 103. *Id.*

Petitioner argues that Patent Owner has not met its burden because it makes no statement that the substitute claims are patentable over prior art not of record, does not include any discussion of the level of ordinary skill in the art, and does not discuss what was previously known regarding the features of the substitute claims. Paper 25 ("Opp."), 2.

We agree that, although it is Patent Owner's burden to show patentability over the prior art, Patent Owner does not assert, or direct us to evidence, that the IDE claimed in the proposed substitute claims was novel over other IDE's known in the art. *See* Tr. 53:15-54:13. Instead, Patent Owner focuses only on Coad, the Oracle Primers, Antis, and U.S. Patent No. 6,785,668 B1 ("Polo"). Accordingly, Patent Owner has not met the burden it undertook by putting forth the proposed amended claims. For that reason, the Motion to Amend is *denied* to the extent it seeks entry of substitute claims 17, 18, and 20-25.

Even if Patent Owner's burden was to show patentability over only the prior art of record, we would not be persuaded that the proposed claims are patentable. To the contrary, we are persuaded that the proposed claims would have been obvious over Coad combined with either the Oracle Primers or Antis.

*1. Proposed Substitute Claim 17*

Claim 17, the proposed substitute for independent claim 1, is identical to original claim 1 except that it adds the following limitation to the visualizer element: "the graphical representations of flows showing a flow within the retrieved source code between data manipulation procedures in an order in which the data manipulation procedures are performed on retrieved data." Second Mot. to Amend 2 (emphasis omitted).

Patent Owner argues that proposed substitute claim 17 is patentable over Coad combined with the Oracle Primers, Coad combined with Antis, and Polo. Mot. to Amend 8-11. According to Patent Owner, the added limitation "clarifies that the flow is 'between' data manipulation procedures and that the data manipulation procedures must be performed on 'retrieved data.'" Mot. to Amend 8-9.

Patent Owner explains that because the graphical depictions of JDBC or embedded SQL in the environment of Coad do not access a database *directly*, they would only show the retrieval of data ("retrieved data"), but would not show a subsequent data manipulation step or flow between subsequent data manipulation steps. *Id.* at 9 (citing Ex. 2001 ¶¶ 67-68). Thus, according to Patent Owner, proposed substitute claim 17 would not have been obvious over the combination of Coad and the Oracle Primers. *Id.* As for the combination of Antis and Coad, Patent Owner argues that neither Coad nor Antis discloses graphical representations of flows of data manipulation procedures being performed on retrieved data. *Id.* at 9-10 (citing Ex. 2001 ¶ 69).

We do not find these arguments persuasive. As discussed above, we have determined that a person of ordinary skill in the art would have found

independent claim 1 obvious over the combination of Coad and the Oracle Primers. We are not persuaded that the added limitation of proposed substitute claim 17 would not have been obvious to a person of skill in the art in view of the disclosure of Coad and the Oracle Primers.

a. *Data Manipulation Procedures*

Patent Owner's arguments all rely on the added limitation requiring that a graphical representation show "a flow within the retrieved source code between data manipulation procedures." Despite the fact that the term "data manipulation procedure" is not used in the '936 patent, Patent Owner does not provide a claim construction for the term. *See* Mot. to Amend. Based on Patent Owner's patentability arguments, we infer that a "data manipulation procedure" under Patent Owner's interpretation requires that the procedure *directly* access data in a database. *See* Paper 26 ("Mot. to Amend Reply") 2-3 ("A function call in source code to an external program is not a 'data manipulation procedure' since any data manipulation would occur *external* to the source code."); Tr. 47:20-24 ("I think that the base assumption there is that a function call within C++ would be a data manipulation procedure, and I think that's entirely inconsistent with the specification read by one skilled in the art in 2001.").

Petitioner argues that this interpretation of "data manipulation procedures" is too narrow. Opp. 5. Instead, Petitioner asserts that the proper interpretation of the claim term is based on the interpretation of "data manipulation language." *Id.* According to Petitioner, because a data manipulation procedure requires only a procedure that is used to access data in a database, but does not require that access to be direct, a "data manipulation procedure" correspondingly is a procedure that accesses data

in a database, and extends to procedures that access the data indirectly. *Id.* Patent Owner appears to agree that the interpretation of "data manipulation procedure" is tied to the interpretation of "data manipulation language." *See* Tr. 49:22-50:9 ("Essentially, once you start defining data manipulation languages as broad as C++ with embedded SQL, you've eviscerated the point of adding this claim limitation."); *see also* Mot. to Amend Reply 3 (referring to "data manipulation procedures" as "DML procedures").

For the reasons explained with respect to the interpretation of "data manipulation language," we agree with Petitioner that the broadest reasonable interpretation of "data manipulation procedure" does not require direct access to a database, and, thus, interpret "data manipulation procedure" to mean a procedure used to access data in a database, such as to retrieve, insert, delete, or modify data in the database.

b. *Retrieved Data*

Again, the term "retrieved data" is not used in the '936 patent and Patent Owner does not proffer a proposed interpretation of this term. However, Patent Owner asserts that data retrieved from a database are no longer "retrieved data" once they are returned to the database. Mot. to Amend Reply 3. Petitioner does not address the construction of this element. We are persuaded that our patentability analysis is unaffected, regardless of whether we adopt Patent Owner's definition. Therefore, we proceed under Patent Owner's understanding of the term.

*c. Patentability*

We are not persuaded that proposed substitute claim 17 is patentable over Coad combined with the Oracle Primers. Patent Owner argues because the "data manipulation procedures are performed on retrieved data," a C++

function call with embedded SQL would not meet the additional claim limitation of proposed substitute claim 17. Mot. to Amend Reply 3; *see also* Tr. 50:10-16. Specifically, Patent Owner asserts that "one skilled in the art would not consider" following an embedded SQL retrieve step with a second embedded SQL manipulation step "because that would be inefficient and unnecessary." Mot. to Amend Reply 3. Patent Owner, however, does not direct us to evidence supporting this attorney argument.

Petitioner, on the other hand, states that "[a]fter retrieving data, additional functional calls may be used to perform data manipulation procedures on the retrieved data." Opp. 6-7 (citing Ex. 1015 ¶ 53). We credit Dr. Roussopoulos's testimony on this issue, which he bases on language in the Oracle8 Primer. Ex. 1015 ¶ 53 (citing Ex. 1013, 93, 95, 103, 118). We, therefore, are not persuaded that proposed substitute claim 17 is patentable over the combination of Coad and the Oracle Primers.

*2. Proposed Substitute Claim 18*

Claim 18, the proposed substitute for dependent claim 3, depends from proposed substitute claim 17 and adds the following limitation to claim 3: "having procedure icons and arrows that show an actual execution path within the retrieved source code as performed on the retrieved data." Second Mot. to Amend 2.

According to Patent Owner, proposed substitute claim 18 is patentable because "[t]he UML diagrams of Coad, at best, disclose all possible pathways within the source code, as opposed to the added limitation that would only show a graphical execution path of the actual flow of data within the source code." Mot. to Amend 12.

We do not agree with Patent Owner's interpretation of the scope of proposed substitute claim 18. The claim does not require, on its face, that *only* the actual execution path be shown. Instead, it simply states that such "actual execution path" will be shown. Thus, Patent Owner concedes that Coad discloses this limitation by showing all possible pathways, which logically includes the actual execution path.

Thus, we are not persuaded that claim 18 is patentable over Coad combined with Antis.

*3. Proposed Substitute Claim 23*

Claim 23, the proposed substitute for dependent claim 8, is written in independent form to include the language of proposed substitute claim 17. Second Mot. to Amend 3-4. In addition, proposed substitute claim 23 adds the following limitations to the visualizer element: "having data processing procedure icons and arrows to depict program flow or icons depicting data processing steps and arrows to depict flow of data in an order data processing procedures or data processing steps occur" and "the data processing procedures or data processing steps being graphically depicted as being completed prior to showing flow to a next procedure or step." *Id.* at 4.

Patent Owner points to Figures 9 and 19 of the original '936 patent application as providing support for the feature that "the data processing procedures or data processing steps being graphically depicted as being completed prior to showing flow to a next procedure or step." Mot. to Amend 7-8. Specifically, Patent Owner asserts that "one skilled in the art would understand that the ['936] patent discloses the graphical depiction of flow between completed DML procedures or steps." Mot. to Amend Reply 5 (citing Ex. 2001 ¶¶ 72-76).

Petitioner argues that proposed substitute claim 23 is unpatentable due to a lack of written description support because the '936 patent specification does not disclose graphical representations including steps that are completed prior to showing flow to the next step. Opp. 13-14. Addressing Mr. Zatkovich's testimony on the issue, Petitioner argues that the testimony describes aspects of the underlying source code from which a flow diagram may be created, but does not point to any particular feature of the figures or any particular language in the '936 patent application that depicts this limitation. *Id.* at 14.

A motion to amend must set forth the support in the original disclosure of the patent for each claim that is added or amended and the support in an earlier-filed disclosure for which the benefit of the filing date of the earlier filed disclosure is sought. 37 C.F.R. § 42.221(b). We agree with Petitioner that Patent Owner's Motion to Amend fails to show where the '936 patent supports the limitation "the data processing procedures or data processing steps being graphically depicted as being completed prior to showing flow to a next procedure or step."

As an initial matter, Patent Owner does not explain what is meant by "graphically depicted as being completed." This language is not found in the specification of the '936 patent, and Patent Owner does not direct us to language in the '936 patent that sheds light on the meaning of this phrase. In support of the Motion to Amend, Patent Owner directs us to the declaration of Mr. Zatkovich. Ex. 2001. Mr. Zatkovich, however, also does not explain what meaning a person of ordinary skill in the art would give to the phrase. Without such explanation, Patent Owner has neither provided a sufficient

explanation of the additional claim language nor established sufficient written description support for such language.

Moreover, in the Motion to Amend, Patent Owner does not identify specifically what portion of the specification actually supports this limitation. *See, e.g.,* Mot. to Amend 7 ("The limitations [of claim 23] are supported by the original specification, page 3, line 23 to page 4, line 2; page 28, lines 1—16; page 29, lines 3-6; and Figs. 9, 17, and 19."). In its Reply, Patent Owner asserts that Figure 19 of the '936 patent shows source code in the bottom of the window "and the corresponding graphical representation of flow between *completed* DML procedures in the top window." Mot. to Amend Reply 5. Patent Owner adds that the specification states that the visualizer window represents "the procedures and data blocks as program flow icons 126." *Id.* (emphasis omitted). Nothing in this language explains how any of the figures of the '936 patent "graphically depict [a procedure] as being completed." Patent Owner directs us to testimony of Mr. Zatkovich stating that Figure 9 depicts a "simple flow" where "[e]ach step is a process that begins when the process receives an[] input and terminates when the outputs are sent to another step." Ex. 2001 ¶ 73. However, we are not persuaded by this testimony that Figure 9 provides sufficient support for this limitation. For example, it is unclear how Figure 9 "graphically depicts [a procedure] as being completed."

Thus, we are not persuaded that claim 23 is patentable over Coad combined with Antis.

*4. Proposed Substitute Claims 20-22, 24, and 25*

Patent Owner argues that proposed substitute claims 20-22, 24, and 25 are patentable over the prior art for the same reason as proposed substitute

claims 17 and 23, and does not present separate arguments as to the alleged patentability of these claims. Mot. to Amend 14. For the reasons discussed above, we are not persuaded by these arguments. Thus, we are not persuaded that Patent Owner has met its burden to show that proposed substitute claims 20-22, 24, and 25 are patentable over the prior art.

## III. CONCLUSION

Petitioner has shown, by a preponderance of the evidence, that the challenged claims are unpatentable based on the following grounds: (1) claim 1 would have been obvious over Coad combined with Oracle Primer and Oracle8 Primer; (2) claims 1, 3, 5, 6, 8, and 10 would have been obvious over Antis combined with Coad; (4) claim 7 would have been obvious over Antis combined with Coad and Eick; and (5) claim 9 would have been obvious over Antis combined with Coad and Building Applications.

Petitioner has not shown that claim 4 is unpatentable. Claims 2 and 11-16 are not at issue in this trial.[3]

Patent Owner has not shown that its proposed substitute claims 17, 18, and 20-25 are patentable over the prior art.

Accordingly, it is

ORDERED that claims 1, 3, and 5-10 of the '936 patent are determined to be *unpatentable*;

FURTHER ORDERED that Patent Owner's Motion to Amend Claims is *denied*; and

[3] In the Decision to Institute, we declined to institute an *inter partes* review of claims 2 and 11-16 because we were not persuaded that Petitioner had shown that there was a reasonable likelihood of prevailing on its challenges to these claims. Dec. 11, 19, 21.

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

PETITIONER:

John Biernacki
David Cochran
John Marlott
Joshua Nightingale
Jones Day
jvbiernacki@jonesday.com
dcochran@jonesday.com
jamarlott@jonesday.com
jrnightingale@jonesday.com

PATENT OWNER:

George Yu
Laura Brutman
James Hanft
Schiff Hardin LLP
gyu@schiffhardin.com
lbrutman@schiffhardin.com
jhanft@schiffhardin.com