# APPELLEE'S BRIEF

CORRECTED

No. 2008-1283

# United States Court of Appeals for the Federal Circuit

THE MATHWORKS, INC.,

*Plaintiff - Appellant,*

v.

COMSOL AB, AND COMSOL, INC.,

*Defendants - Appellees.*

Appeal from the United States District Court
for the Eastern District of Texas,
In Case No. 6:06-CV-00334, Judge Leonard Davis

## Brief of Appellees



Thomas H. Watkins
State Bar No. 20928000
Elizabeth G. Bloch
State Bar No. 02495500
BROWN MCCARROLL, L.L.P.
111 Congress Avenue, Suite 1400
Austin, Texas 78701
(512) 479-9746
(512) 479-1101 (fax)

*ATTORNEYS FOR DEFENDANTS-APPELLEES,*
*COMSOL AB AND COMSOL, INC.*

No. 2008-1283

# United States Court of Appeals for the Federal Circuit

THE MATHWORKS, INC.,

*Plaintiff - Appellant,*

v.

COMSOL AB, AND COMSOL, INC.,

*Defendants - Appellees.*

Appeal from the United States District Court
for the Eastern District of Texas,
In Case No. 6:06-CV-00334, Judge Leonard Davis

**Brief of Appellees**

Thomas H. Watkins
State Bar No. 20928000
Elizabeth G. Bloch
State Bar No. 02495500
BROWN MCCARROLL, L.L.P.
111 Congress Avenue, Suite 1400
Austin, Texas 78701
(512) 479-9746
(512) 479-1101 (fax)

*ATTORNEYS FOR DEFENDANTS-APPELLEES, COMSOL AB AND COMSOL, INC.*

## CERTIFICATE OF INTEREST

Counsel for Defendants-Appellees COMSOL AB and COMSOL, Inc. hereby certify the following:

1. The full name of every party represented by me is:

   COMSOL AB
   COMSOL, Inc.

2. The names of the real parties in interest (if the party named in the caption is not the real party in interest) represented by me are:

   None.

3. All parent corporations and any publicly held companies that own 10% or more of the stock of the party or amicus curiae represented by me are:

   None.

4. The names of all law firms and the partners or associates that appeared for the party or amicus now represented by us in the trial court or agency or are expected to appear in this Court are:

   NIXON PEABODY, LLP: Maia Harris, Gina McCreadie, Nicholas G. Papastavros, Mark D. Robins, Marc S. Kaufman

   CAPSHAW DERIEUX, LLP: Elizabeth L. DeRieux

   LYNN TILLOTSON & PINKER, LLP: Michael P. Lynn, Richard A. Smith

   BROWN MCCARROLL, LLP: Thomas H. Watkins, Elizabeth G. Bloch, Albert A. Carrion

## TABLE OF CONTENTS

## TABLE OF AUTHORITIES

### CASES

**MISCELLANEOUS**



## STATEMENT OF RELATED CASES

There is currently no appeal in or from this same civil action in this Court or in any other court of appeals. Counsel is not aware of any case pending is this or any other court that will directly affect or be directly affected by this Court's decision herein.

## TABLE OF ABBREVIATIONS

| | |
|---|---|
| ***Parties*** | |
| MathWorks | Plaintiff-Appellant The MathWorks, Inc. |
| COMSOL | Defendants-Appellees COMSOL AB and COMSOL, Inc., collectively |
| ***Defined Terms*** | |
| The '338 patent or the patent | U.S. Letters Patent No. 7,051,338, issued May 23, 2006, assigned to MathWorks, and entitled "Method and System for Accessing Externally-Defined Objects from an Array-Based Mathematical Computing Environment" (A0019-0031) |
| (____:____) | Column and line number in the '338 patent |
| (A____) | Joint Appendix page(s) |
| COMSOL Script® 1.x | Accused software made, used, offered for sale, sold, and/or imported by COMSOL |
| PTO | Patent and Trademark Office |
| District court | United States District Court for the Eastern District of Texas, Honorable Leonard Davis presiding |
| Court | United States Court of Appeals for the Federal Circuit |
| MATLAB® | Software created by MathWorks; the commercial embodiment of the '338 patent |

## STATEMENT OF THE ISSUE

COMSOL agrees with the first part of MathWorks' Statement of the Issue, but disagrees with its argumentative statement about the district court's construction. The issue is simply whether the district court construed the "rank" and "ranking" terms of the '338 patent correctly.

## STATEMENT OF THE FACTS

MathWorks' MATLAB software program provides a technical computing environment for solving mathematical problems, designing complex mechanical and electrical control systems, and performing statistical analysis, among other things. (A0026 1:13-25). The purported invention embodied in the '338 patent allows a user in MATLAB environments to interface a separate, object-oriented computing environment such as Java, in order to access and use operations or functions predefined in that environment.

In object-oriented programming languages, such operations or functions are called "methods." Methods are associated with objects—modules of computer code that specify the data types of a data structure and the methods that can be applied to it. Objects are the building blocks used to design computer programs, and each object is a unique instance of its corresponding class. (See, A0027 3:40-43). A class defines the characteristics (including the methods) common to all objects of a certain kind. (A0027 3:37-39). Methods can be invoked (executed) by

generating a request to invoke the method when it is desired to perform a specific function on a data structure. The request generally will specify the method by the name. *See* David Flannigan, *Java In A Nutshell*, 3rd Ed., O'Reilly, 1999, pp. 43, 59-61. When there is only one method with the requested method name, then that method will be invoked in response to the request.

At times, however, multiple methods will have the same method name, but they are distinguishable by their number of expected input parameters or their data types. This is known as "overloading." Overloading will resolve to an appropriate method based on (for example) an analysis of the different methods' data types. Overloading was well known prior to the filing of the '338 patent application in computer languages and environments such as Java and C++. *See* Flannigan, *Java In a Nutshell*, p. 61.

The '338 patent concerns the issue of method overloading when a user in a technical computing environment (such as MATLAB) seeks to invoke a method from an object-oriented computing environment (such as Java). To address the issue, the purported invention disclosed and claimed in the '338 patent compares data types passed in a request to invoke a method from the technical computing environment with the data types of each identically named method referenced in the request in order to determine the suitability of such method to receive the input parameters passed by the request.

As claim 1 of the '338 patent illustrates, once suitability has been determined, the purported invention then "ranks" the method signatures based on the suitability of each method to receive the input parameters passed in the requested invocation from the technical computing environment to determine which of the overloaded methods in the object-oriented should be selected. The method with the highest ranking will then be selected for invocation. (See, e.g., A0028 6:18-22; A0026 2:5-6). This avoids, or at least minimizes, the need for converting data types of the input parameters passed in the method invocation to data types of the method that is invoked.

This case involves the construction of what will be referred to collectively as the "ranking terms" found in claim 1 and claim 15 of the '338 patent. MathWorks contended that the ranking terms should be construed to mean "assigning to a particular class," and that there were two possible classes: suitable and unsuitable. The district court, however, agreed with COMSOL and construed these terms as follows:

| | |
|---|---|
| "ranking the method signatures" | Placing the method signatures in an ordered manner relative to one another |
| "rank[s] the method signatures" | Place[s] the method signatures in an ordered manner relative to one another |
| "the ranking" | The list of method signatures placed in an ordered manner relative to one another |

The district court acknowledged that multiple method signatures could have an equal rank on the list. (A0010).

The '338 patent also includes the separate notion of the "fitness ranking" of method signatures as part of the ranking process. The "fitness ranking" is a calculation for each method signature that is "representative of a level of suitability of the data types of the input parameters of the method represented by the signature to use the input parameters passed by the requested method invocation." (A0030 9:18-22). Whereas the "ranking" of the method signatures is a list that places the method signatures in an ordered manner relative to one another, the "fitness ranking" is a separate calculation for each of the method signatures on that list, to determine its *level* of suitability in an absolute, as opposed to relative, sense.

## SUMMARY OF THE ARGUMENT

MathWorks proffers a construction of the "ranking" terms in the '338 patent that is not consistent with the ordinary meaning of the terms, is contrary to the way in which those terms are used in the claims and in the specification, and is not supported by MathWorks' own use of those terms in its prosecution of the patent. MathWorks now wants to be its own lexicographer in defining the "ranking" terms, yet it failed to provide notice to the public of those special definitions in the '338 patent.

In contrast, the district court's construction of the ordinary meaning of the "ranking" terms is entirely consistent with the intrinsic record, including the specification, the use of those terms in the claims themselves, and the prosecution history. It is MathWorks' proposed construction, not the district court's, that violates the doctrine of claim differentiation by not giving effect to all of the claimed terms in the patent. MathWorks' construction of "ranking" as assigning to a class of suitable or unsuitable fails to differentiate and renders superfluous the "comparing" element of the claim, which is the step "to determine suitability." And the court's construction does not, as MathWorks asserts, exclude the preferred embodiment.

## ARGUMENT AND AUTHORITIES

### I. MathWorks' proposed construction is not supported by the language of the claims, the specification, or the prosecution history.

MathWorks' proposed construction of "ranking" is "assigning to a particular class." (MathWorks' Brief at 12). MathWorks states that "ranking" is "the process of evaluating and characterizing a methods' suitability," and that "ranking" is nothing more than assigning to one of two classes: suitable or unsuitable. (MathWorks' Brief at 8, 18). Under MathWorks' construction, "all that is necessary for 'ranking' of method signatures is a determination of suitability (or not) and an assignment to a corresponding class." (MathWorks' Brief at 12). In fact, as discussed below, MathWorks continues to insist that the assignment can be

"random," which renders the claimed step of ranking meaningless, even the construction proferred by MathWorks.

MathWorks' proposed construction is wrong for several reasons. First, it ignores and renders superfluous another portion of the claim, which states that the "comparing" step is the one that "determine(s) suitability." Second, MathWorks' proposed construction is contrary to the ordinary meaning of the term "ranking" and is not otherwise defined in the patent. Finally, there is nothing in the claims or the specification that supports MathWorks' proposed construction.

### A. MathWorks' proposed construction is contrary to the language of the claims, and renders part of the claim meaningless.

As this Court has noted, it is a "bedrock principle" of patent law that the claims of a patent define the invention. *Phillips v. AWH Corporation,* 415 F.3rd 1303, 1312 (Fed. Cir. 2005). Courts should first look to the words of the claims themselves to define the scope of the patented invention. *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996). The words of the claims at issue here are as follows:

> 1. A method for invoking a method defined with an object-oriented computing environment comprising:
>
> retrieving a set of method signatures for a method referenced in a requested method invocation, where each method signature corresponds to a method provided by an object within an object-oriented environment, and further wherein each signature includes a method name

and lists any data types of input parameters to be received by the corresponding method;

**comparing** the data types of input parameters of each method represented by the signatures to data types of input parameters passed by the requested method invocation **to determine suitability** of each method to receive input parameters passed by the requested method invocation;

**ranking** the method signatures **based on the determined suitability** of each method represented by the signatures to receive the input parameters passed by the requested method invocation;

selecting one of the method signatures according to the ranking; and

invoking, in response to the requested method invocation, the method of the object-oriented computing environment corresponding to the selected method signature; wherein the request method invocation is requested by an array-based computing environment provided by a mathematical tool.

(A0029 8:56-67-A0030 9:1-15) (emphasis added).

15. A computer program, tangibly stored on a computer-readable medium, for invoking a method defined within an object-oriented environment, the computer program comprising instructions operable to cause a programmable processor to:

retrieve a set of method signatures for a method referenced in a requested method invocation, where each method signature corresponds to a method provided by an object within an object-oriented environment, and further wherein each signature includes a method name and a data type for each input parameter received by the corresponding method;

> **compare** the data types of each input parameter of each method represented by the signatures to data types of input parameters passed by the requested method invocation **to determine suitability** of each method to receive the input parameters passed by the requested method invocation;
>
> **rank** the method signatures **based on the determined suitability** of each method represented by the signatures to receive the input parameters passed by the requested method invocation;
>
> select one of the method signatures according to the ranking; and
>
> invoke, in response to the requested method invocation, the method of the object-oriented computing environment corresponding to the selected method signature; wherein the request method invocation is requested by an array-based computing environment provided by a mathematical tool.

(A0030 10:12-42) (emphasis added).

In both of these claims there are five separate elements or functions: retrieving a set of method signatures; comparing the data types to determine suitability; ranking the method signatures based on the determined suitability; selecting one of the method signatures according to the ranking; and invoking the appropriate method. Under MathWorks' proposed construction, however, "ranking" is the element that determines suitability and categorizes a method as either suitable or not. This necessarily ignores the "comparing" element, which the claim language itself indicates is the one "to determine suitability," not "ranking." MathWorks' proposed construction thus violates two basic tenants of claims

construction. It fails to construe the term "ranking" in the context of the entire patent, including the rest of the claim itself, and it renders meaningless or superfluous an element expressly recited as part of the claim.

MathWorks' construction of "ranking," although not common or ordinary, may be plausible in a vacuum, but not in the context of the remaining language of the patent. A word must be construed "not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification." *Phillips,* 415 F.3d at 1313. Here, the proper context requires a reading of the "ranking" terms in the context of the other elements of the claims as well. This clearly indicates that "comparing . . . to determine suitability" is different from "ranking . . . based on the determined suitability."

Claims should be interpreted with an eye toward giving effect to all of the terms in the claim. *Bicon, Inc. v. Straumann Co.*, 441 F.3d 945, 950 (5th Cir. 2006). MathWorks' proposed construction of "ranking," however, fails to give effect to the term "comparing," which is described in the claim itself as the element that is used "to determine suitability." If, as MathWorks suggests, the "ranking" element is the one that determines suitability and assigns to one of two classes—suitable or not suitable—then the comparing element is rendered superfluous. A claim construction that would render a claimed term or a phrase superfluous is to be avoided. *Merck & Company v. Teva Pharms. USA, Inc.*, 395 F.3d 1364, 1372

(Fed. Cir. 2005) ("A claim construction that gives meaning to all the terms of a claim is preferred over one that does not do so").

### B. "Assigning to a class" is neither an ordinary meaning of "ranking" nor a specific definition given to the term in the patent.

Words contained in a claim "are generally given their ordinary and customary meaning." *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996). Moreover, the ordinary and customary meaning of a claim term is the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention. *Innova/Pure Water, Inc. v. Safari Water Filtration Systems, Inc.*, 381 F.3d 1111, 1116 (Fed. Cir. 2004). A patentee is free to be his or her own lexicographer, but any special definition that is given to a word, other than its ordinary meaning, must be clearly set forth in the specification. *Intellicall, Inc. v. Phonemetrics*, 952 F.2d 1384, 1388 (Fed. Cir. 1992); *Vitronics*, 90 F.3d at 1582.

Here, MathWorks provided no special definition in the patent, either expressly or by implication. *See Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 979 (Fed. Cir. 1995) (*en banc*), *aff'd*, 116 S.Ct. 1384 (1996) (the specification acts as a dictionary when it expressly defines terms used in the claims or when it defines terms by implication). MathWorks asserts that "ranking" means "assigning to a class," but nowhere in the '338 patent is "ranking" defined in this manner, nor

is such definition implied by the context or use of the term in either the claims or the specification.

It was therefore appropriate, as discussed in more detail below, for the district court to determine the ordinary and commonly understood meaning of the "ranking" terms.

### C. There is nothing in the specification that supports MathWorks' proposed construction.

Claims "must be read in view of the specification, of which they are a part." *Markman*, 52 F.3d at 979. As long as a limitation is not improperly read into the claims, the specification is the primary basis for construing the claims. *Phillips*, 415 F.3d at 1315-16.

MathWorks proposes a strained and insupportable reading of the specification to suggest that it confirms that the "ordinary understanding of 'ranking'. . . [is to] categorize, or assign, the methods to a class based upon suitability." (MathWorks' Brief at 15). But the specification does not say this, either expressly or impliedly.

A fatal flaw in MathWorks' reasoning is revealed by its suggestion that the selection based on "ranking" is not necessarily based on the highest ranking (even though this is what the preferred embodiment indicates), but rather the tool might select "alphabetically" or "at random." (MathWorks' Brief at 23). MathWorks does not and cannot explain how it is even possible to "select . . . according to the

ranking" without having an ordered ranking. The specification clearly teaches that the purpose of the invention is to select the "best" method, not a random one:

- "Based on this comparison, the invention selects a method that best accepts the input arrays." (A0019 Abstract).
- "Best Fit." (A0023 Fig. 3).
- "Based on this comparison, the invention automatically selects a method that best accepts the input arrays." (A0026 2:5-6).
- ". . . signature selector automatically selects a method from object-oriented environment that is best able receive (sic) the data from the array inputs." (A0027 4:25-28).

Since the selection of the signature is recited in the claims as being "based on the ranking," a random selection would not result in achieving the objectives of the invention stated in the specification and noted above. A random selection would render the other claimed steps (retrieving, comparing, and ranking) useless. In short, there is nothing in the specification that supports MathWorks' construction, and in fact, the specification indicates that its construction is wrong.

## II. The district court's construction was correct.

### A. It was proper to use the ordinary meaning of the "ranking" terms since the patent did not provide a different definition.

As noted above, a patentee may provide a special definition of a word that is different from the ordinary meaning, but that special definition must be clearly set forth in the specification. Otherwise, resorting to the terms' ordinary meaning is appropriate. *See Symantec Corp. v. Computer Associates International, Inc.*, 522

F.3d 1279, 1291 (Fed. Cir. 2008) ("because the specification does not reveal any special definition for the terms [at issue], we must construe those terms according to their ordinary meaning").

This Court has noted that reference to sources such as dictionaries may be appropriate "so long as the dictionary definition does not contradict any definition found in or ascertained by a reading of the patent documents." *Phillips*, 415 F.3d at 1322-23; *see also Mangosoft, Inc. v. Oracle Corp.*, 525 F.3d 1327, 1330 (Fed. Cir. 2008) ("even *Phillips* recognized that reference to [dictionaries] is not prohibited so long as the ultimate construction given to the claims in question is grounded in the intrinsic evidence and not based upon definitions considered in the abstract.").

This case is similar to *Cat Tech LLC v. Tubemaster, Inc.*, 528 Fd.3d 871 (Fed. Cir. 2008), where construction of the term "spacing" was at issue. This Court discerned "no persuasive evidence that the word 'spacing' has a specially defined meaning in the field of art encompassed by the [patent]." *Id.* This Court further reasoned that the ordinary and customary meaning of the term by those of ordinary skill and the art involved little more than "the application of the widely accepted meaning of commonly understood words." *Id.* (citing *Phillips*, 415 F.3d at 1314). Resorting to a dictionary definition of the term "spacing" was thus appropriate.

Here, although the district court did not specifically reference a dictionary in its Memorandum Opinion, its construction of the "ranking" terms is consistent with that found in standard English dictionaries:

- "ranking *n* – an act or instance of indicating relative standing." *Random House Webster's Unabridged Dictionary,* 2nd Ed., 1998;
- "rank *n* – an orderly arrangement." *Webster's New 20th Century Dictionary 2nd Edition,* 1977, at 1493;
- "rank" *v* – to place in a rank or ranks." Id.

These definitions are consistent with that found in the *Wikipedia* entry for "rank," which states that the term "is usually related to a relative position or to some kind of ordering." http://en.wikipedia.org/wiki/Rank (last accessed July 16, 2008). Since MathWorks did not specifically define the "ranking" terms in its '338 patent, and since there is no evidence that someone skilled in the relevant art would understand the terms to mean something different, the dictionary would be an appropriate source for the ordinary meaning of these terms.

### B. The district court's construction does not violate the doctrine of claim differentiation.

MathWorks suggests that the district court's construction of the "ranking" terms violates the doctrine of claim differentiation because it does not differentiate that term from the term "fitness ranking" found in the dependent claims. That argument is wrong, since "fitness ranking" still has its own meaning separate and apart from "ranking." In any event, the doctrine of claim differentiation is not a

hard and fast rule, but rather a presumption that can be overcome; it cannot be used to broaden claims beyond their correct scope. *Seachange, International, Inc. v. C-Cor Inc.*,413 F.3d 1361 (Fed. Cir. 2005).

Under the district court's construction, "ranking" means "placing the method signatures in an ordered manner relative to one another." It is a list of the method signatures placed in order. "Fitness ranking," in contrast, is not a list. As recited in dependent claim 2, a fitness ranking is calculated for "each method signature;" in other words, there is a separate fitness ranking for each method signature on a one-to-one basis. The fitness ranking is a designation, such as a classification, of fitness of a particular method signature, not an ordered list of all the method signatures. In fact, it is MathWorks' proposed definition of "ranking" that would violate the doctrine of claim differentiation. MathWorks proffers that ranking is assigning a classification to each individual method signature. Yet this is precisely what "fitness ranking" is, as recited in the dependent claims.

In addition, the language of the two claims further indicates their differences. "Ranking" is "based on the determined **suitability of each method** represented by the signatures **to receive the input parameters** passed by the requested method invocation." (A0030 9:4-7) (emphasis added). "Fitness ranking" is "representative of a **level of suitability of the data types** of the input parameters of the method represented by the signature **to use the input**

**parameters** passed by the requested method invocation." (A0030 9:18-22) (emphasis added).

"Fitness ranking" thus calculates the absolute "*level* of suitability" of the "*data types*" to "*use* the input parameters," for each method signature, things that the "ranking" does not necessarily do. (A0030 9:18-22). The ranking puts the method signatures that have been determined to be suitable in an order relative to one another, but it does not necessarily provide information about how suitable they may be in an absolute context—the "level" of suitability. For example, the top ranked method signature may have a low level of suitability, or a low ranked method signature may have a high level of suitability.

To use an analogy, students receiving the top five highest raw scores on a bar exam would be the top five ranked students. The ranking of these students lists them in order relative to one another. The actual raw scores of each of those top five students, however, may nonetheless all be low, indicating that none has a very high level of suitability, or fitness, to practice law. In other words, being at the top of a ranking means simply that of the group tested, that student had the highest score relative to the others of that group. If that student's score were 100, it would indicate a high level of suitability, and thus a high fitness ranking. On the other hand, if the highest ranked student scored only a 72, then even though that student

had a high ranking relative to others in the group, that student is not highly suited to be an attorney; i.e. does not have a high "fitness ranking" in the absolute sense.

The district court's construction recognized and gave effect to the different meanings of the term "ranking" and "fitness ranking," and thus did not violate the doctrine of claim differentiation.

### C. The district court properly construed the "ranking" terms in the context of the order of the steps.

As this court has noted:

> Unless the steps of a method actually recite an order, the steps are not ordinarily construed to require one. However, such a result can ensue when the method steps implicitly require that they be performed in the order written.

*Interactive Gift Express, Inc. v. CompuServe, Inc.*, 256 F.3d 1323, 1342-43 (Fed. Cir. 2000). The two-part test set forth in *Interactive Gift* requires the court first to look at the claim language to determine if, as a matter of logic or grammar, they must be performed in the order written. *Id.* at 1343. If not, courts will look to the rest of the specification to determine whether it "directly or implicitly requires such a narrow construction." *Id.* at 1343.

Here, the first part of the test resolves the question, since the claim language itself indicates that the steps must be performed in their written order. *See Loral Fairchild Corp. v. Sony Electronics Corp.*, 181 F.3d 1313, 1321 (Fed. Cir. 1999); *Mantech Environmental Corp. v. Hudson Environmental Services, Inc.*, 152 F.3d

1368, 1375-76 (Fed. Cir. 1998) (holding that the steps of a method claim had to be performed in their written order because each subsequent step referenced something logically indicating the prior step had been performed). That is precisely the case here.

The first ordered step of the claim retrieves a set of method signatures. The next step takes that particular set of retrieved method signatures and compares the data types of input parameters of each method, in order to determine suitability. The comparing step can only happen after a set of method signatures had been retrieved; otherwise there is nothing to compare. The ranking step involves "ranking of method signatures *based on the determined suitability* of each method." (A0030 9:4-7) (emphasis added). This ranking step can only take place once suitability has been already determined in the previous step, since the ranking is based on that determination. The final two steps are selecting and invoking, which also necessarily, by logic, grammar, and common sense, occur in order after the first three steps.

The district court was correct in construing the "ranking" terms in the context of the ordered steps described in the claims. This context, together with the rule against rendering a claim term superfluous, indicates that the "comparing" step, which determines suitability, comes before the "ranking," which ranks the method signatures based on that determination. It is MathWorks' construction, not

the district court's, that fails "to appreciate the differences between two method steps." (MathWorks' Brief at 27).

### D. The district court's construction does not exclude the preferred embodiment.

MathWorks suggests that the district court's construction excludes the preferred embodiment in the '338 patent because it is allegedly based on the assumption that multiple signatures are necessarily retrieved, while in the preferred embodiment, according to MathWorks, the "set" of method signatures retrieved may contain zero, one, or multiple method signatures. (MathWorks' Brief at 30). That portion of the patent that MathWorks cites for this proposition, however, is not designated in the patent as the "preferred embodiment," and it does not indicate that the "set" retrieved may contain zero, one, or multiple method signatures. That portion of the patent states simply, "according to the technique, a list of method signatures corresponding to a particular class and method name is retrieved from the object-oriented environment." (A0026 2:13-15).

There is nothing in what MathWorks describes as the preferred embodiment or anywhere else in the patent that indicates zero or one may be retrieved, and the use of the plural "method signatures" indicates that more than one will be retrieved. Indeed, this portion of the patent goes on to state, "based on the ranking, one of the method signatures is selected and the corresponding method within the object-oriented environment is invoked unless no suitable method is found, in

which case an error condition is raised." (A0026 2:22-25) (emphasis added). This clearly indicates that one of multiple method signatures will be selected, and that if none is found, an error results. In the case of an error, the remaining steps of ranking, selecting, and invoking are not performed. Therefore, the example set forth by MathWorks, while conceivable, is not consistent with the claimed invention.

MathWorks also complains that the district court's construction requires multiple signatures to be given a different rank, but that is simply not the case. Indeed, the district court recognized that an ordinal ranking "allows items to have an equal rank." (A0010). After the comparison to determine suitability, two or more method signatures may be ranked the same based on that suitability.

**E. The prosecution history supports the district court's construction.**

The prosecution history of a patent is a proper source for determining the proper construction of a term. *Phillips*, 415 F.3d at 1314 (citing *Innova*, 381 F.3d at 1116). The two statements mentioned by the district court in its Memorandum Opinion are entirely consistent with the words of the claims themselves and the specification in that MathWorks' invention selects the best method signature based on the ranking of the method signatures in order of suitability. During prosecution, MathWorks used the term "rank[s] [ranking]" to require an ordinal relationship between the ranked items: "the ranking referred to in step 3 [of claim 1] is the

ranking of the method signatures *in order* of suitability for handling the input parameters from the array-based computing environment." (A0014) (emphasis added). And "the methods are ranked based on which methods can best accept the input parameters of the data from the calling array-based computing environment." (A0014, quoting A0364).

MathWorks cannot now, in the context of litigation, recapture coverage that was surrendered during prosecution of the patent based on the representations made to the PTO. Read as a whole, in the context of the entire patent, these statements in the prosecution history are entirely consistent with and confirm the district court's construction of the disputed terms.

## CONCLUSION AND PRAYER

The district court's construction of the "ranking" terms in the '338 patent are correct in all respects, and MathWorks has offered no plausible argument for adopting its constrained reading of the words in the claims at issue. In the context of the claims and the specification, the "ranking" terms can only mean placing the method signatures in an ordered manner relative to one another. MathWorks failed to provide any other definition in the patent, and cannot now attempt to adopt one that it did not disclose, particularly since it is inconsistent with its own use of the terms in the prosecution of the patent.

COMSOL therefore requests that this Court affirm the judgment below in all respects. COMSOL requests such other relief to which it may be entitled.

July 22, 2008

Respectfully submitted,

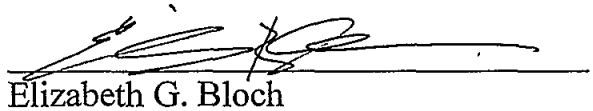
Thomas H. Watkins by egb with permission
Elizabeth G. Bloch
Brown McCarroll, L.L.P.
111 Congress Avenue, Suite 1400
Austin, Texas 78701
(512) 479-9746
(512) 479-1101 (Facsimile)

*Attorneys for Defendants-Appelles, COMSOL AB and COMSOL, Inc.*

## CERTIFICATE OF SERVICE

I hereby certify that on July 22, 2008, two bound copies of the foregoing BRIEF OF DEFENDANTS-APPELLEES were served by overnight mail through a third-party commercial carrier (Federal Express) upon the following principal counsel:

Gregory A. Castanias
Jones Day
51 Louisiana Avenue, N.W.
Washington, D.C. 2001-2113



Elizabeth G. Bloch

## CORRECTED CERTIFICATE OF SERVICE

I hereby certify that on July 29, 2008, two bound copies of the foregoing CORRECTED BRIEF OF DEFENDANTS-APPELLEES were served by overnight mail through a third-party commercial carrier (Federal Express) upon the following principal counsel:

Gregory A. Castanias
Jones Day
51 Louisiana Avenue, N.W.
Washington, D.C. 2001-2113

Elizabeth G. Bloch

## CERTIFICATE OF COMPLIANCE

1. This brief complies with the page limitation of Federal Rule of Appellate Procedure 32(a)(7)(A), because it contains less than 30 pages.

2. This brief complies with the typeface requirements of Federal Rule of Appellate Procedure 32(a)(5)(A), and the type style requirements of Federal Rule of Appellate Procedure 32(a)(6), because it has been prepared in a proportionally spaced typeface using Microsoft Word XP in Times New Roman 14 point font.

Dated: July 22, 2008 Respectfully submitted,

Elizabeth G. Bloch

# ADDENDUM

**IN THE UNITED STATES DISTRICT COURT**
**FOR THE EASTERN DISTRICT OF TEXAS**
**TYLER DIVISION**

| | | |
|---|---|---|
| The MathWorks. Inc., | ) | |
| | ) | |
| Plaintiff. | ) | |
| | ) | CASE NO. 6:06-cv-334 LED |
| v. | ) | |
| | ) | JURY TRIAL DEMANDED |
| COMSOL AB and COMSOL, Inc., | ) | |
| | ) | |
| Defendants. | ) | |
| | ) | |

**FINAL JUDGMENT**

THIS MATTER having come before the Court, and the plaintiff The MathWorks, Inc. ("MathWorks"), and the defendants COMSOL AB and COMSOL, Inc. (collectively, "COMSOL") having entered into a Stipulation regarding the claims and counterclaims in this action, IT IS HEREBY ORDERED, ADJUDGED AND DECREED AS FOLLOWS:

1. This Court has jurisdiction over the parties and the subject matter of this action.

2. MathWorks is a Delaware corporation with its principal place of business in Natick, Massachusetts.

3. COMSOL AB is a Swedish corporation with its principal place of business in Stockholm, Sweden.

4. COMSOL, Inc. is a Massachusetts corporation with its principal place of business in Burlington, Massachusetts.

5. COMSOL has stipulated that COMSOL Script 1.0b infringes claims 1 - 4, 11 – 17, 19 – 20, and 22 of United States Patent No. 7,051,338 ("the '338 patent"). Although COMSOL has not stipulated to infringement of COMSOL Script 1.0 and 1.0a. COMSOL consents to an injunction as to COMSOL Script 1.0 and 1.0a as well.

6. Based on the Court's construction of "ranking the method signatures," "rank[s] the method signatures," and "the ranking," memorialized in its February 13, 2008 Memorandum Opinion, MathWorks has stipulated that COMSOL Script 1.1 and 1.2 do not infringe the claims asserted, which are 1, 12-15, and 22 of the '338 patent. MathWorks' stipulation will not bind MathWorks if the claim construction ruling is reversed, vacated or otherwise modified.

7. This Court hereby enjoins COMSOL and its officers, agents, servants, employees, attorneys, and other persons who are in active concert or participation with them, in the United States and its territories and possessions, for the life of claims 1 - 4, 11 – 17, 19 – 20, and 22 of the '338 patent, from manufacturing, using, selling, offering for sale, promoting, distributing, displaying in any medium or otherwise disposing of in any manner in the United States, and from importing or causing importation into the United States, COMSOL Script 1.0, 1.0a, and 1.0b.

8. This Court hereby dismisses COMSOL Counterclaim Count I for Declaratory Judgment of Non-Infringement with prejudice as to COMSOL Script 1.0, 1.0a, and 1.0b, and without prejudice as to COMSOL Script 1.1 and 1.2.

9. This Court hereby dismisses COMSOL Counterclaim Count II for Declaratory Judgment of Invalidity without prejudice.

10. The parties shall bear their own attorneys' fees and costs.

**So ORDERED and SIGNED this 10th day of March, 2008.**

**LEONARD DAVIS**
**UNITED STATES DISTRICT JUDGE**

**IN THE UNITED STATES DISTRICT COURT**
**FOR THE EASTERN DISTRICT OF TEXAS**
**TYLER DIVISION**

| | | |
|---|---|---|
| The MathWorks, Inc., | ) | |
| | ) | |
| Plaintiff, | ) | |
| | ) | CASE NO. 6:06-cv-334 LED |
| v. | ) | |
| | ) | JURY TRIAL DEMANDED |
| COMSOL AB and COMSOL, Inc., | ) | |
| | ) | |
| Defendants. | ) | |
| | ) | |

**STIPULATION**

Plaintiff The MathWorks, Inc. ("MathWorks") and Defendants COMSOL AB and COMSOL, Inc. ("COMSOL") (collectively the "Parties") hereby stipulate and agree as follows:

1. COMSOL hereby stipulates that COMSOL Script 1.0b infringes claims 1 - 4, 11 – 17, 19 – 20, and 22 of United States Patent No. 7,051,338 ("the '338 patent").

2. Based on the Court's construction of "ranking the method signatures," "rank[s] the method signatures," and "the ranking," memorialized in its February 13, 2008 Memorandum Opinion, MathWorks hereby stipulates that COMSOL Script 1.1 and 1.2 do not infringe the claims asserted, which are 1, 12-15, and 22 of the '338 patent. MathWorks intends to appeal the claim construction ruling, and the stipulation in this paragraph is not intended to bind MathWorks if the ruling is reversed, vacated or otherwise modified.

3. COMSOL consents to an injunction whereby COMSOL and its officers, agents, servants, employees, attorneys, and other persons who are in active concert or participation with them are enjoined in the United States and its territories and possessions, for the life of claims 1 - 4, 11 – 17, 19 – 20, and 22 of the '338 patent, from manufacturing, using, selling, offering for sale, promoting, distributing, displaying in any medium or otherwise disposing of in any manner

in the United States, and from importing or causing importation into the United States, COMSOL Script 1.0b (due to paragraph 1 above) and, although not stipulating to infringement of COMSOL Script 1.0 and 1.0a, consents to this injunction as to COMSOL Script 1.0 and 1.0a.

4. COMSOL agrees that its Counterclaim Count I for Declaratory Judgment of Non-Infringement shall be dismissed with prejudice as to COMSOL Script 1.0, 1.0a, and 1.0b, and shall be dismissed without prejudice as to COMSOL Script 1.1 and 1.2.

5. COMSOL agrees that its Counterclaim Count II for Declaratory Judgment of Invalidity shall be dismissed without prejudice.

6. Each party agrees to bear its own attorneys' fees and costs.

7. A [Proposed] Final Judgment reflecting this Stipulation is attached as Exhibit A.

IT IS SO STIPULATED.

Dated: March 7, 2008

*/s/ Krista S. Schwartz*
Terence M. Murphy (14707000)
Attorney-in-Charge
Email: tmmurphy@jonesday.com
Thomas R. Jackson (10496700)
Email: trjackson@jonesday.com
JONES DAY
2727 North Harwood Street
Dallas, TX 75201-1515
Telephone: (214) 220-3939
Facsimile: (214) 969-5100

Krista S. Schwartz (06238053)
Email: ksschwartz@jonesday.com
JONES DAY
77 West Wacker
Chicago, IL 60601-1692
Telephone: (312) 782-3939
Facsimile: (312) 782-8585

Mark N. Reiter (16759900)
GIBSON, DUNN & CRUTCHER
2100 McKinney Ave., Suite 1100
Dallas, Texas 75201
Telephone: (214) 698-3360
Facsimile: (214) 571-2907

Carl Roth (17312000)
Brendan C. Roth (24040132)
Email: br@rothfirm.com
Amanda A. Abraham (24055077)
Email: aa@rothfirm.com
LAW OFFICES OF CARL ROTH
115 N. Wellington
Marshall, Texas 75670
Telephone: (903) 935-1665
Facsimile: (903) 935-1797

**Attorneys for Plaintiff**
**THE MATHWORKS, INC.**

*/s/ Thomas H. Watkins (by permission)*
Thomas H. Watkins
State Bar No. 20928000
Scott R. Kidd
State Bar No. 11385500
BROWN McCARROLL, L.L.P.
111 Congress Avenue, Suite 1400
Austin, Texas 78701
Telephone: (512) 472-5456
Facsimile: (512) 479-1101

Elizabeth L. DeRieux
State Bar No. 05770585
BROWN MCCARROLL, L.L.P.
1127 Judson Road, Suite 220
P.O. Box 3999 (75606-3999)
Longview, Texas 75601-5157
Telephone: (903) 236-9800
Facsimile: (903) 236-8787

Mark D. Robins (admitted *pro hac vice*)
Nicholas G. Papastavros (admitted *pro hac vice*)
Maia H. Harris (admitted *pro hac vice*)
Gina M. McCreadie (admitted *pro hac vice*)
NIXON PEABODY LIP
100 Summer Street
Boston, Massachusetts 02110
Telephone: 617.345.1000
Facsimile: 617.345.1300

Michael P. Lynn, P.C.
State Bar No. 12738500
Richard A. Smith
State Bar No. 24027990
LYNN, KILOTON & PINKER, L.L.P.
750 North St. Paul Street, Suite 1400
Dallas, Texas 75201
Telephone: 214.981.3800
Facsimile: 214.981.3839

**Attorneys for Defendants**
**COMSOL AB and COMSOL, INC.**

**CERTIFICATE OF SERVICE**

The undersigned hereby certifies that all counsel of record who are deemed to have consented to electronic service are being served with a copy of this document via the Court's CM/ECF system per Local Rule CV-5(a)(3). Any other counsel of record will be served by facsimile transmission and/or first class mail this 7th day of March, 2008.

*/s/ Krista S. Schwartz*

**IN THE UNITED STATES DISTRICT COURT**
**FOR THE EASTERN DISTRICT OF TEXAS**
**TYLER DIVISION**

| | | |
|---|---|---|
| **THE MATHWORKS, INC.** | § | |
| | § | |
| **Plaintiff** | § | |
| | § | |
| **vs.** | § | **CASE NO. 6:06CV334** |
| | § | **PATENT CASE** |
| **COMSOL AB and COMSOL, INC.** | § | |
| | § | |
| **Defendants** | § | |

**MEMORANDUM OPINION**

This Memorandum Opinion construes the terms in United States Patent No. 7,051,338 (the "'338 Patent").

**BACKGROUND**

The '338 Patent issued on May 23, 2006 and claims a method and apparatus that facilitate invoking object methods defined within an object-oriented environment from an array-based technical computing environment, the type of environment often used in conventional mathematical tools. The invention first retrieves a set of method signatures from object methods, where each method signature includes the method's name and lists the data types of the method's input parameters. The invention then compares the data types of each method's input parameters to the data types of the input parameters the array-based environment will pass to the method. The invention executes this comparison to determine the suitability of each method to receive the input parameters from the array-based environment. Based upon the determined suitability of each method, the invention ranks the method signatures and selects one method signature according to the ranking. Finally, the invention invokes the method that corresponds to the selected method signature.

The MathWorks, Inc. ("MathWorks") alleges Comsol AB and Comsol, Inc. (collectively, "Comsol") infringe various claims of the '338 Patent. At the *Markman* hearing, counsel for Comsol agreed to MathWorks's construction for the remaining disputed terms except for "rank[s] [ranking] the method signatures" and "the ranking."

**APPLICABLE LAW**

"It is a 'bedrock principle' of patent law that 'the claims of a patent define the invention to which the patentee is entitled the right to exclude.'" *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc) (quoting *Innova/Pure Water Inc. v. Safari Water Filtration Sys., Inc.*, 381 F.3d 1111, 1115 (Fed. Cir. 2004)). In claim construction, courts examine the patent's intrinsic evidence to define the patented invention's scope. *See id.*; *C.R. Bard, Inc. v. U.S. Surgical Corp.*, 388 F.3d 858, 861 (Fed. Cir. 2004); *Bell Atl. Network Servs., Inc. v. Covad Commc'ns Group, Inc.*, 262 F.3d 1258, 1267 (Fed. Cir. 2001). This intrinsic evidence includes the claims themselves, the specification, and the prosecution history. *See Phillips*, 415 F.3d at 1314; *C.R. Bard, Inc.*, 388 F.3d at 861. Courts give claim terms their ordinary and accustomed meaning as understood by one of ordinary skill in the art at the time of the invention in the context of the entire patent. *Phillips*, 415 F.3d at 1312–13; *Alloc, Inc. v. Int'l Trade Comm'n*, 342 F.3d 1361, 1368 (Fed. Cir. 2003).

The claims themselves provide substantial guidance in determining the meaning of particular claim terms. *Phillips*, 415 F.3d at 1314. First, a term's context in the asserted claim can be very instructive. *Id.* Other asserted or unasserted claims can also aid in determining the claim's meaning because claim terms are typically used consistently throughout the patent. *Id.* Differences among the claim terms can also assist in understanding a term's meaning. *Id.* For example, when a dependent claim adds a limitation to an independent claim, it is presumed that the independent claim does not include the limitation. *Id.* at 1314–15.

"[C]laims 'must be read in view of the specification, of which they are a part.'" *Id.* (quoting *Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 979 (Fed. Cir. 1995) (en banc)). "[T]he specification 'is always highly relevant to the claim construction analysis. Usually, it is dispositive; it is the single best guide to the meaning of a disputed term.'" *Id.* (quoting *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996)); *Teleflex, Inc. v. Ficosa N. Am. Corp.*, 299 F.3d 1313, 1325 (Fed. Cir. 2002). This is true because a patentee may define his own terms, give a claim term a different meaning than the term would otherwise possess, or disclaim or disavow the claim scope. *Phillips*, 415 F.3d at 1316. In these situations, the inventor's lexicography governs. *Id.* Also, the specification may resolve ambiguous claim terms "where the ordinary and accustomed meaning of the words used in the claims lack sufficient clarity to permit the scope of the claim to be ascertained from the words alone." *Teleflex, Inc.*, 299 F.3d at 1325. But, "'[a]lthough the specification may aid the court in interpreting the meaning of disputed claim language, particular embodiments and examples appearing in the specification will not generally be read into the claims.'" *Comark Commc'ns, Inc. v. Harris Corp.*, 156 F.3d 1182, 1187 (Fed. Cir. 1998) (quoting *Constant v. Advanced Micro-Devices, Inc.*, 848 F.2d 1560, 1571 (Fed. Cir. 1988)); *see also Phillips*, 415 F.3d at 1323. The prosecution history is another tool to supply the proper context for claim construction because a patent applicant may also define a term in prosecuting the patent. *Home Diagnostics, Inc., v. Lifescan, Inc.*, 381 F.3d 1352, 1356 (Fed. Cir. 2004) ("As in the case of the specification, a patent applicant may define a term in prosecuting a patent.").

Although extrinsic evidence can be useful, it is "'less significant than the intrinsic record in determining the legally operative meaning of claim language.'" *Phillips*, 415 F.3d at 1317 (quoting *C.R. Bard, Inc.*, 388 F.3d at 862). Technical dictionaries and treatises may help a court understand the underlying technology and the manner in which one skilled in the art might use claim terms, but

technical dictionaries and treatises may provide definitions that are too broad or may not be indicative of how the term is used in the patent. *Id.* at 1318. Similarly, expert testimony may aid a court in understanding the underlying technology and determining the particular meaning of a term in the pertinent field, but an expert's conclusory, unsupported assertions as to a term's definition is entirely unhelpful to a court. *Id.* Generally, extrinsic evidence is "less reliable than the patent and its prosecution history in determining how to read claim terms." *Id.*

**RANK**

Claims 1, 2, 15, and 16 contain the terms "rank[s] [ranking] the method signatures" and "the ranking." MathWorks contends "rank[s] [ranking] the method signatures" means "to assign to a particular class the method signatures" and "the ranking" means "assignment to a particular class." Comsol contends "rank[s] [ranking] the method signatures" means "place[s] [placing] the method signatures in an ordered manner relative to one another" and "the ranking" means "the list of method signatures placed in an ordered manner relative to one another."

The ordinary meaning of "rank" requires an ordinal relationship between ranked items and allows items to have an equal rank. The claims use the terms "rank[s] [ranking]" and "the ranking" in accordance to their ordinary meanings and require the "rank[s] [ranking]" step to do more than simply determine whether a method is "suitable" or "unsuitable."

The claims require a determination of each method's suitability before execution of the "rank[s] [ranking]" step. Courts generally do not construe method claims to require the method be performed in the order written. *Altiris, Inc. v. Symantec Corp.*, 318 F.3d 1363, 1369 (Fed. Cir. 2003) (quoting *Interactive Gift Express, Inc. v. Compuserve Inc.*, 256 F.3d 1323, 1342–43 (Fed. Cir. 2001)). However, courts limit a method claim to cover only methods performed in the order written if the method steps actually recite an order. *Id.* If the method steps do not actually recite an order,

courts may limit a method claim to cover only methods performed in the order written if the method steps implicitly require that they be performed in the order written. *Id.*

Method steps implicitly require performance in the order written in two instances. First, method steps implicitly require sequential performance if the claim language, as a matter of logic, requires the steps be performed in the order written. *Id.* at 1369–70. (citing *Interactive Gift*, 256 F.3d at 1343). Second, if, as a matter of logic, the claim language does not require the steps be performed in the order written, method steps implicitly require sequential performance if the specification "directly or implicitly requires such a narrow construction." *Id.* at 1370 (quoting *Interactive Gift*, 256 F.3d at 1343).

Claim 1 claims, in part, a method that comprises the following steps: "retrieving a set of method signatures for a method referenced in a requested method invocation"; "comparing the data types of input parameters of each method represented by the signatures to data types of input parameters passed by the requested method invocation to determine suitability of each method to receive input parameters passed by the requested method invocation"; "ranking the method signatures based on the determined suitability of each method represented by the signatures to receive the input parameters passed by the requested method invocation"; and "selecting one of the method signatures according to the ranking." '338 Patent, col. 8:56–col. 9:15. Claim 15, which claims a computer program comprising instructions operable to cause a programmable processor to perform a method, contains similar limitations. *Id.* at col. 10:12–43.

The method steps do not recite an actual order. However, logic requires performance of the "comparing" and "ranking" steps in the order written. The method requires "comparing . . . to determine suitability of each method" and "ranking the method signatures based on the determined suitability of each method." The "comparing" step "determine[s] suitability" and thus requires

performance before the claimed method can "rank[] the method signatures based on the determined suitability."

As the "comparing" and "ranking" steps must be performed in the order written, claim 1 indicates the "ranking" step is not so broad to cover assignment of method signatures to a "suitable" or "unsuitable" class because the claimed method has already determined suitability. *Hyperion Solutions Corp. v. OutlookSoft Corp.*, 422 F. Supp. 2d 760, 772 (E.D. Tex. 2006) (Ward, J.) (rejecting both parties' proposed constructions and noting "[b]edrock principles of claim construction counsel against a construction that renders additional limitations superfluous") (citing *Merck & Co., Inc. v. Teva Pharma. USA, Inc.*, 395 F.3d 1364, 1372 (Fed. Cir. 2005); *see also Primos, Inc. v. Hunter's Specialties, Inc.*, 451 F.3d 841, 847–48 (rejecting proposed construction of claim term where proposed construction would render another claim term superfluous). As such, the "ranking" step must do more than assign each method signature a "suitable" or "not suitable" ranking.

The parties additionally dispute whether the doctrine of claim differentiation applies and requires a broad construction of "rank[s] [ranking]" and "the ranking." Courts presume a difference in meaning and scope when a patentee uses different phrases in separate claims. *Phillips*, 415 F.3d at 1314–15. Where a party seeks to limit an independent claim with language that appears in a dependant claim, the presumption is especially strong. *Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 910 (Fed. Cir. 2004). However, the doctrine of claim differentiation is not a "hard and fast rule," and courts cannot use the doctrine to broaden claims beyond their correct scope, determined in light of the intrinsic record and relevant extrinsic evidence. *Seachange Int'l, Inc. v. C-COR, Inc.*, 413 F.3d 1361, 1369 (Fed. Cir. 2005); *see also Phillips*, 415 F.3d at 1312–15.

Claim 2 claims the method of claim 1 "wherein the ranking the method signatures comprises

calculating a fitness ranking representative of a level of suitability of the data types of the input parameters of the method represented by the signature to use the input parameters passed by the requested method invoked." '338 Patent, col. 9:16–21. Claim 16, which depends on claim 15, contains an identical limitation. *Id.* at col. 10:43–49.

The parties agree "fitness ranking" requires no construction. Claims 2 and 16 themselves define the term, as the "fitness ranking" is "representative of a level of suitability of each of the data types of the input parameters of the method represented by the signature to use the input parameters passed by the requested method invocation." *Id.* at col. 9:16–21, col. 10:43–49. Mathworks contends the doctrine of claim differentiation requires a broader construction of "rank[ing]" and "the ranking," as "fitness ranking" is a narrower form of "the ranking."

The doctrine of claim differentiation does not mandate a broad construction of "rank[ing]" and "the ranking." Claims 2 and 16 narrow claims 1 and 15 on the basis that the "rank[ing]" step comprises "calculating a fitness ranking." The doctrine of claim differentiation presumes the "rank[ing]" steps in claim 1 and 15 are broader than "calculating a fitness ranking for each method signature, the fitness ranking representative of a level of suitability of each of the data types of the input parameters of the method represented by the signature to use the input parameters passed by the requested method invocation."

The requirement of an ordinal relationship amongst ranked items is different than "calculating a fitness ranking . . . representative of a level of suitability of each of the data types of the input parameters of the input parameters of the method represented by the signature to use the input parameters passed by the requested method invocation." Thus, the doctrine of claim differentiation does not broaden the terms "rank[ing]" and "the ranking" beyond their ordinary meanings.

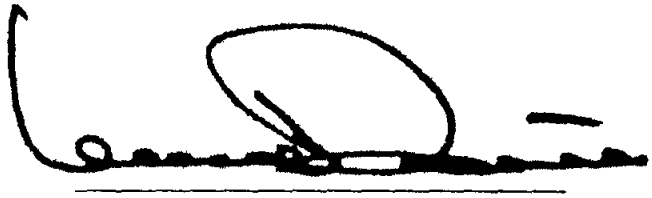
The remainder of the intrinsic evidence supports a construction of "rank[s] [ranking]" that requires an ordinal relationship between the ranked items. The specification broadly discloses a system that ranks method signatures based upon a comparison of the method signatures' data types with the data types the array-based computing environment will pass to a selected method within the object-oriented environment. *Id.* at col. 2:11–49. Similarly, during prosecution, the applicant used the term "rank[s] [ranking]" to require an ordinal relationship between the ranked items. Comsol's Claim Construction Brief, Ex. 5 at TMW-PAT 00000336 ("The ranking referred to in step 3 [of claim 1] is the ranking of the method signatures in order of suitability for handling the input parameters from the array-based computing environment."); *id.* at TMW-PAT 00000279, TMW-PAT 00000333 ("The methods are ranked based on which methods can best accept the input parameters of the data from the calling array-based computing environment.").

The '338 Patent uses the terms "rank[s] [ranking]" and "the ranking" consistent with their ordinary meanings. Thus, "ranking the method signatures" means "placing the method signatures in an ordered manner relative to one another," "rank[s] the method signatures" means "place[s] the method signatures in an ordered manner related to one another," and "the ranking" means "the list of method signatures placed in an ordered manner relative to one another."

## CONCLUSION

For the foregoing reasons, the Court interprets the claim language in this case in the manner set forth above. For ease of reference, the Court's claim interpretations are set forth in a table as Appendix B. The claims with the disputed terms in bold are set forth in Appendix A.

**So ORDERED and SIGNED this 12th day of February, 2008.**



**LEONARD DAVIS**
**UNITED STATES DISTRICT JUDGE**

**APPENDIX A**

**U.S. Pat. No. 7,051,338**

1. A method for invoking a method defined with an object-oriented computing environment comprising:
retrieving a set of method signatures for a method referenced in a requested method invocation, where each method signature corresponds to a method provided by an object within an object-oriented environment, and further wherein each signature includes a method name and lists any data types of input parameters to be received by the corresponding method;
comparing the data types of input parameters of each method represented by the signatures to data types of input parameters passed by the requested method invocation to determine suitability of each method to receive input parameters passed by the requested method invocation;
**ranking the method signatures** based on the determined suitability of each method represented by the signatures to receive the input parameters passed by the requested method invocation;
selecting one of the method signatures according to **the ranking**;
and invoking, in response to the requested method invocation, the method of the object-oriented computing environment corresponding to the selected method signature;
wherein the request method invocation is requested by an array-based computing environment provided by a mathematical tool.

2. The method of claim 1, wherein **ranking the method signatures** comprises calculating a fitness ranking for each signature, the fitness ranking representative of a level of suitability of the data types of the input parameters of the method represented by the signature to use the input parameters passed by the requested method invocation.

3. The method of claim 2, wherein calculating a fitness ranking for each signature includes generating a preference value for each data type of the signature and adjusting the fitness ranking of the corresponding signature as a function of the comparison.

4. The method of claim 2, wherein calculating a fitness ranking for each signature includes calculating a difference in a number of dimensions between the signature data type and the input parameter received from the computing environment.

11. The method of claim 1, wherein comparing each data type of the signature to the data type of the corresponding input parameter includes accessing a data structure storing data types of the object-oriented environment ordered by preference.

12. The method of claim 1, wherein invoking the method includes: converting the input parameters to data types supported by the object-oriented environment; and converting return values from the method to data types supported by the computing environment.

13. The method of claim 1, wherein the object-oriented environment includes a virtual machine, and further wherein invoking the method includes interpreting the method via the virtual machine.

14. The method of claim 1, wherein each signature includes a method name comprising the name of the method in the requested method invocation, and wherein each method represented by the signature corresponds to a method provided by the same object.

15. A computer program, tangibly stored on a computer-readable medium, for invoking a method defined within an object-oriented environment, the computer program comprising instructions operable to cause a programmable processor to:
retrieve a set of method signatures for a method referenced in a requested method invocation, where each method signature corresponds to a method provided by an object within an object-oriented environment, and further wherein each signature includes a method name and a data type for each input parameter received by the corresponding method;
compare the data types of each input parameter of each method represented by the signatures to data types of input parameters passed by the requested method invocation to determine suitability of each method to receive the input parameters passed by the requested method invocation;
**rank the method signatures** based on the determined suitability of each method represented by the signatures to receive

the input parameters passed by the requested method invocation;
select one of the method signatures according to **the ranking**;
and invoke, in response to the requested method invocation, the method of the object-oriented computing environment corresponding to the selected method signature;
wherein the request method invocation is requested by an array-based computing environment provided by a mathematical tool.

16. The computer program of claim 15, wherein the computer program **ranks the method signatures** by calculating a fitness ranking for each signature, the fitness ranking representative of a level of suitability of the data types of the input parameters of the method represented by the signature to use the input parameters passed by the requested method invocation.

17. The computer program of claim 16, wherein the computer program calculates a fitness ranking for each signature by generating a preference value for each data type of the signature and adjusting the fitness ranking of the corresponding signature as a function of the comparison.

19. The computer program of claim 15, wherein the computer program calculates a fitness-ranking for each signature by calculating a difference in a number of dimensions between the signature data type and the input parameter received from the computing environment.

20. The computer program of claim 15, wherein the computer program compares each data type of the signature to the data type of the corresponding input parameter includes by accessing a data structure storing data types of the object-oriented environment ordered by preference.

22. The computer program of claim 15, wherein each signature includes a method name comprising the name of the method in the requested method invocation, and wherein each method represented by the signature corresponds to a method provided by the same object.

**APPENDIX B**

| Ref. Nos. | Term or Phrase to be Construed (Claims) | Court's Construction |
|---|---|---|
| 1 | ranking the method signatures (claim 1, 2)<br><br>rank[s] the method signatures (claim 15, 16)<br><br>the ranking (claim 1, 15) | placing the method signatures in an ordered manner relative to one another<br><br>place[s] the method signatures in an ordered manner related to one another<br><br>the list of method signatures placed in an ordered manner relative to one another |
| 2 | fitness ranking / fitness-ranking (claims 2, 3, 4, 16, 17, 19) | AGREED – *no construction required* |
| 3 | array-based computing environment (claims 1, 15) | AGREED – computing environment in which the data types are primarily represented as arrays of at least two dimensions |
| 4 | mathematical tool (claims 1, 15) | AGREED – *no construction required* |
| 5 | data type(s) (claims 1, 3, 4, 11, 12, 15, 16, 17, 19, 20) | AGREED – category of data characterized by a set of values and operations that can be applied to them |
| 6 | method (claims 1, 2, 12, 13, 14, 15, 16, 22) | AGREED operation or procedure associated with an object |
| 7 | object (claims 1, 11, 14, 15, 22) | AGREED – modules of computer code that specify the data types of a data structure, and the methods that can be applied to the data structure |
| 8 | signature(s) / method signature(s) (claim 1, 2, 3, 4, 11, 14, 15, 16, 17, 19, 20, 22) | AGREED – representation of the method's name and the number and types of parameter(s) of the method |
| 9 | object-oriented environment / object-oriented computing environment (claims 1, 12, 13, 15, 20) | AGREED – a computing environment, such as Java, that supports code defined as objects |

US007051338B1

(12) **United States Patent**
**Foti et al.**

(10) **Patent No.: US 7,051,338 B1**
(45) **Date of Patent: May 23, 2006**

(54) **METHOD AND SYSTEM FOR ACCESSING EXTERNALLY-DEFINED OBJECTS FROM AN ARRAY-BASED MATHEMATICAL COMPUTING ENVIRONMENT**

(75) Inventors: **David A. Foti**, Ashland, MA (US); **Charles G. Nylander**, Merrimack, NH (US)

(73) Assignee: **The MathWorks, Inc.**, Natick, MA (US)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/518,287**

(22) Filed: **Mar. 3, 2000**

(51) **Int. Cl.**
*G06F 9/00* (2006.01)

(52) **U.S. Cl.** ..... 719/328; 719/330

(58) **Field of Classification Search** ..... 709/315; 719/320, 330, 328
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,341,478 A * 8/1994 Travis et al. ..... 709/203

(Continued)

OTHER PUBLICATIONS

David M. Gay, Symlolic-Algebraic Computations in a Modeling Language for Mathematical Programming, Nov. 1999, Schloss Dagstuhl, p. 4-7.*

Hartmut Pohlheim, Genetic and Evolutionary Algorithm Toolbox for use with MATLAB, Jul. 1997. John W.Eaton, A High-Level Interactive Language for Numeriacal Computaions Edition 3 for Octave Verdion 2.1.x, Feb. 1997.*

Nec Corp, Index implementation method for object oriented database—involves comparing value for structure type member variable to obtain size related rank for variables, Oct. 17, 1997.*

Cantin, International Business Machine, Corporation, Persiten Object-Mapping in an Object-Oriented Environment, Mar. 1, 1996 Venners, Eternal Math, 1996.*

Tierman, "An Efficient Search Algorithm to Find the Elementary Circuits of a Graph", *Comm. of the ACM*, .13:722-726, (1970).

Tarjan, "Depth-First Search and Linear Graph Algorithms", SIAM *J. Comp.*. 1:146-160, Jun., (1972).

Tarjan, "Enumeration of the Elementary Circuits of a Directed Graph" *Cornell University Technical Report TR* 72-145 (1972).

IBM Technical Disclosure Bulletin, Generating Event Adapters to Facilitate Connections Between Java Beans, Jan. 1, 1998, p. 1-3.

John Henry Moore, Microsoft's New, Improved Proxy Server, Dec. 1997, p. 1-3.

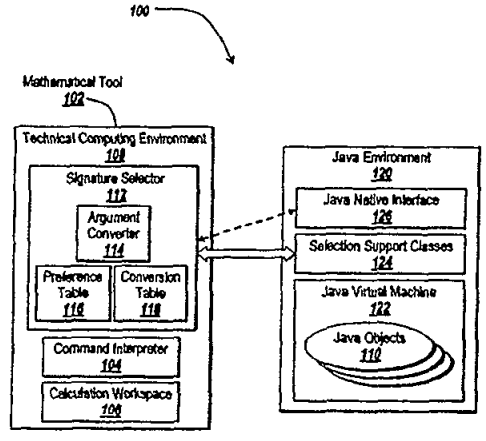
Samir B. Gehani, A Java Based Framework for Explicitly Partitioning, 1997, Section 3.1 Java Beans.

*Primary Examiner*—William Thomson
*Assistant Examiner*—LeChi Truong
(74) *Attorney, Agent, or Firm* Lahive & Cockfield, LLP

(57) **ABSTRACT**

A method and apparatus, including a computer program apparatus, which facilitate invoking methods of objects defined within an object-oriented environment from an array-based technical computing environment often used in conventional mathematical tools. When a method is invoked from the computing environment, the techniques automatically compare the array input parameters with data-types accepted by methods defined within the object-oriented environment. Based on this comparison, the invention selects a method that best accepts the input arrays. The invention, therefore, allows a user to easily invoke methods from external objects, such as Java objects, directly from the technical computing environment of the mathematical tool.

**35 Claims, 5 Drawing Sheets**



U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 6,061,721 A | 5/2000 | Ismael et al. | 709/223 |
| 6,230,160 B1 | 5/2001 | Chan et al. | 707/103 X |
| 6,282,699 B1 * | 8/2001 | Zhang et al | 717/109 |
| 6,289,395 B1 | 9/2001 | Apte et al. | 709/318 |

* cited by examiner

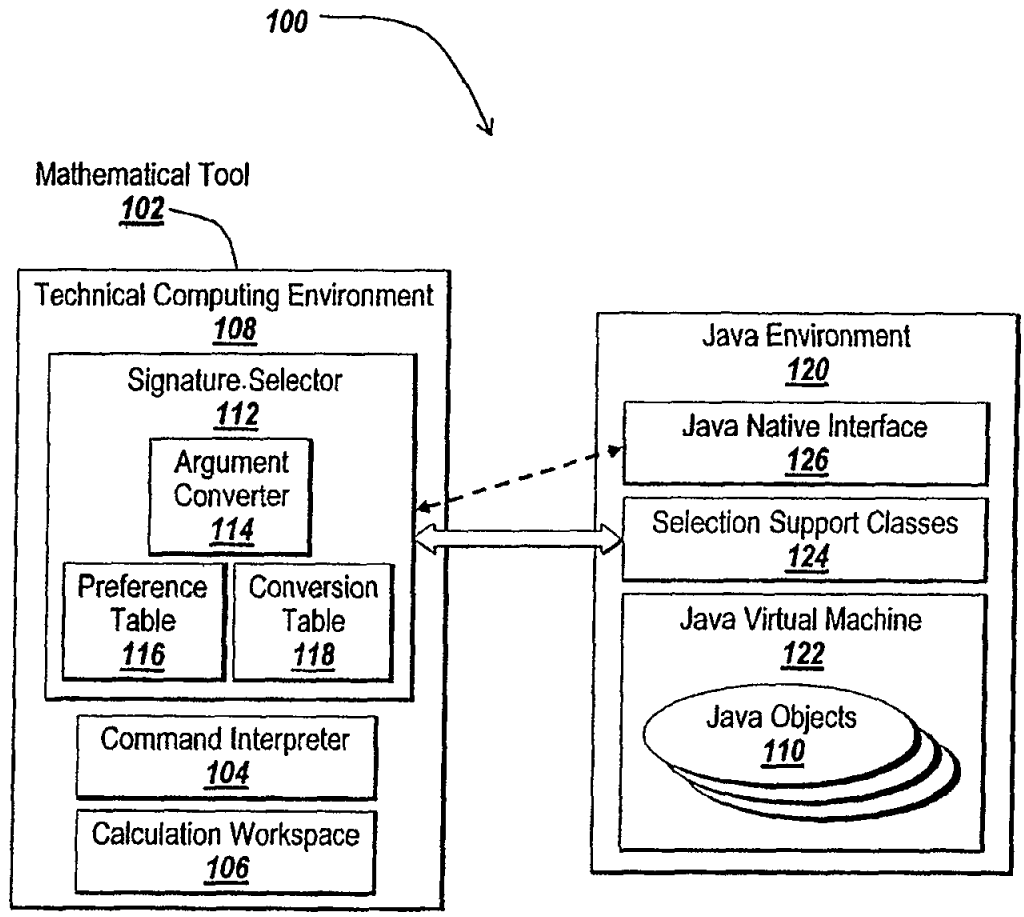

100
Mathematical Tool
102
Technical Computing Environment
108
Signature.Selector
112
Argument Converter
114
Preference Table
116
Conversion Table
118
Command Interpreter
104
Calculation Workspace
106
Java Environment
120
Java Native Interface
126
Selection Support Classes
124
Java Virtual Machine
122
Java Objects
110

*Fig. 1*

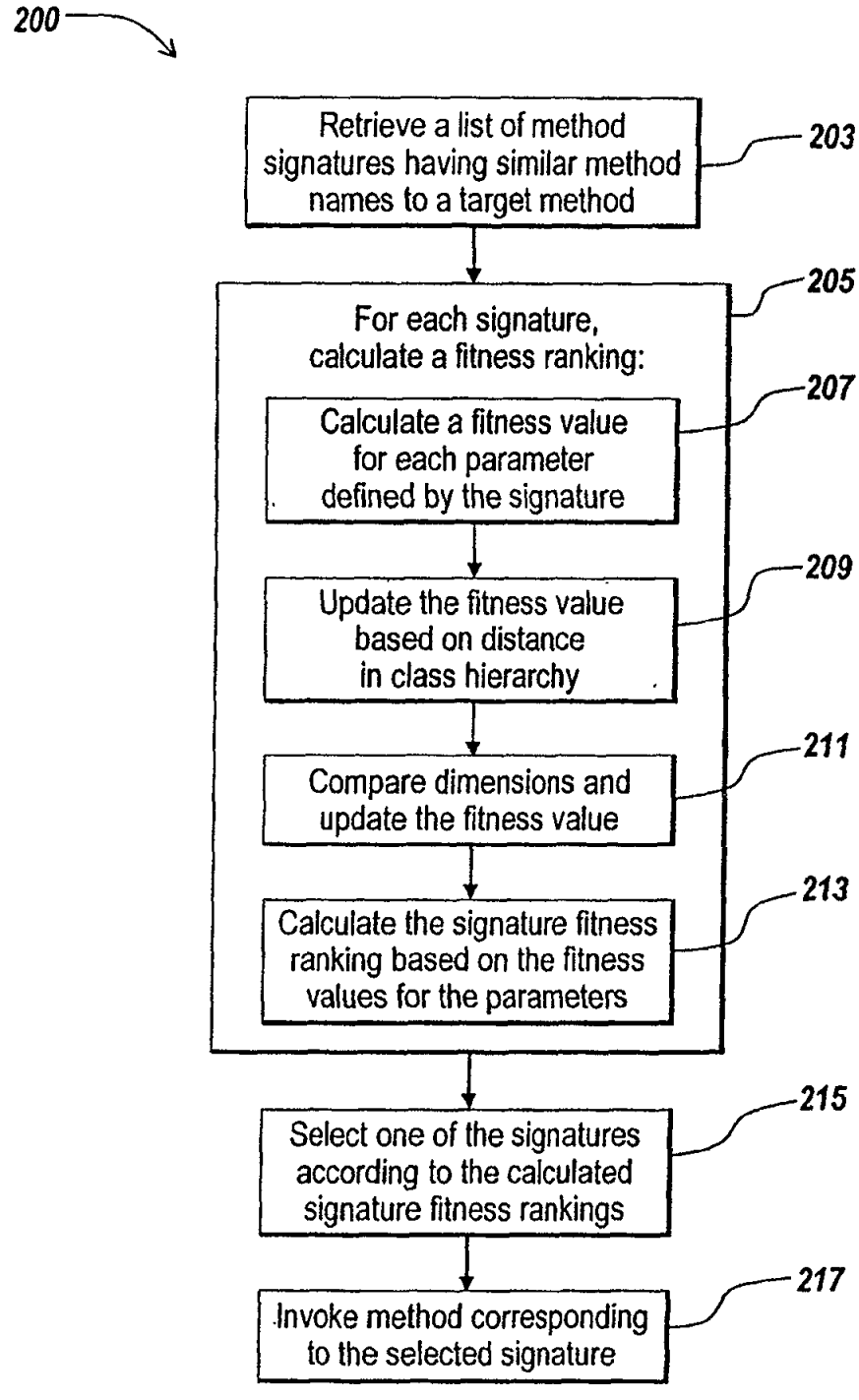

200
Retrieve a list of method signatures having similar method names to a target method
203
For each signature, calculate a fitness ranking:
205
Calculate a fitness value for each parameter defined by the signature
207
Update the fitness value based on distance in class hierarchy
209
Compare dimensions and update the fitness value
211
Calculate the signature fitness ranking based on the fitness values for the parameters
213
Select one of the signatures according to the calculated signature fitness rankings
215
Invoke method corresponding to the selected signature
217

*Fig. 2*

116

| Mathematical Tool Data Type | Java Data Type | | | | | | |
|---|---|---|---|---|---|---|---|
| 310 | Best Fit | | | | | | Worst Fit |
| logical | boolean | | | | | | |
| double | double | float | long | int | short | byte | boolean |
| single | float | double | | | | | |
| char | String | char | | | | | |
| array of char arrays | String | | | | | | |
| array of arrays | Object | | | | | | |
| unsigned 8-bit int | byte | short | int | long | float | double | |
| unsigned 16-bit int | short | int | long | float | double | | |
| unsigned 32-bit int | int | long | float | double | | | |
| signed 8-bit int | byte | short | int | long | float | double | |
| signed 16-bit int | short | int | long | float | double | | |
| signed 32-bit int | int | long | float | double | | | |
| java | Object | | | | | | |

*Fig. 3*

118

| Java Data Type | Mathematical Tool Data Type (for scalar Java types) | Mathematical Tool Data Type (for array Java types) |
|---|---|---|
| boolean | double precision floating point | boolean |
| byte | double precision floating point | 8-bit signed integer |
| char | char | char (16-bit) |
| short | double precision floating point | 16-bit signed integer |
| int | double precision floating point | 32-bit signed integer |
| long | double precision floating point | double precision floating point |
| float | double precision floating point | double precision floating point |
| double | double precision floating point | double precision floating point |
| java.lang.String Object | char array | char array |
| Java object | Reference to Java object | Reference to Java object |

*Fig. 4*

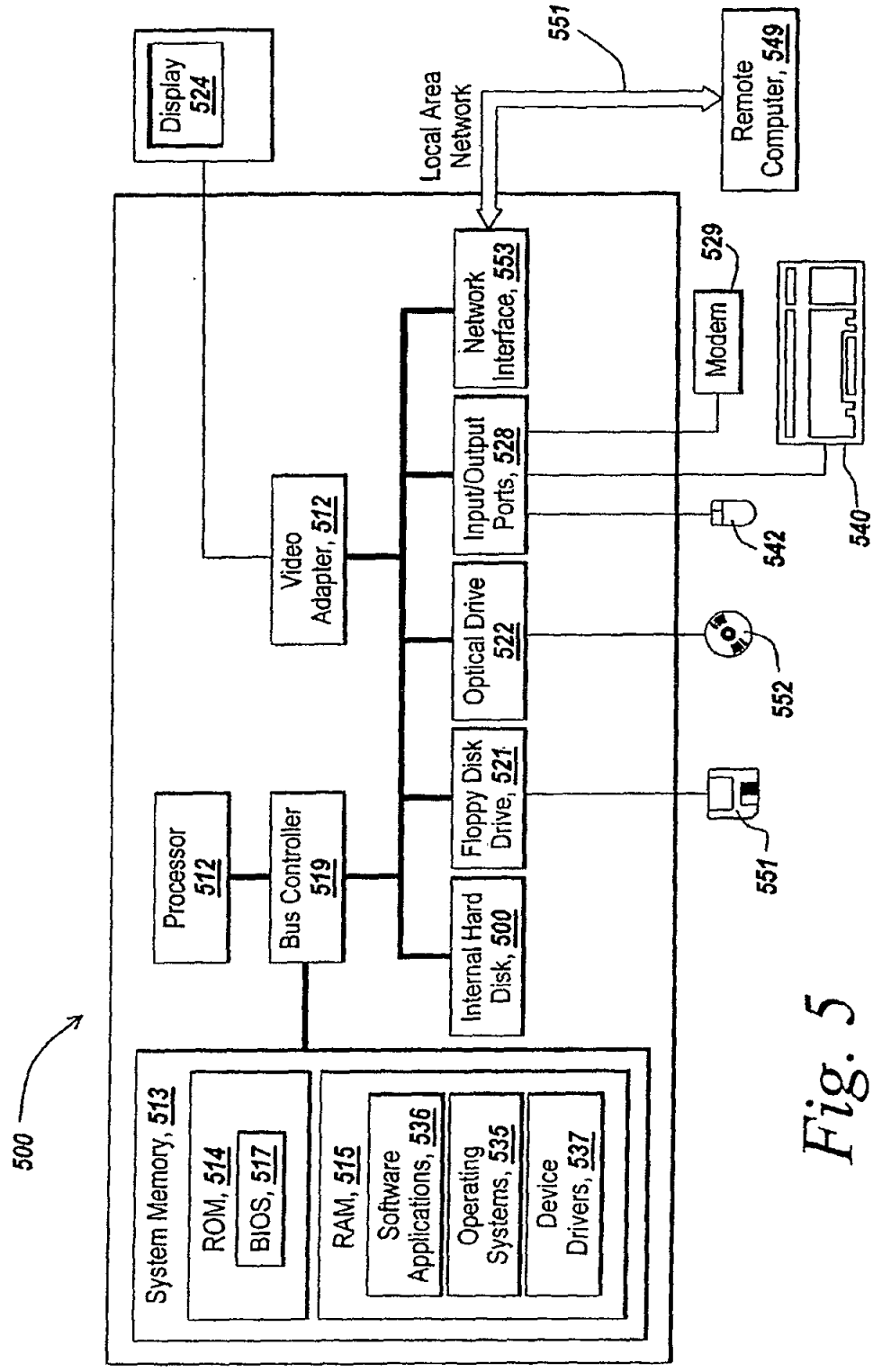

500
System Memory, 513
ROM, 514
BIOS, 517
RAM, 515
Software Applications, 536
Operating Systems, 535
Device Drivers, 537
Processor 512
Bus Controller 519
Internal Hard Disk, 500
Floppy Disk Drive, 521
Optical Drive 522
Input/Output Ports, 528
Network Interface, 553
Video Adapter, 512
Display 524
Local Area Network
551
Remote Computer, 549
Modem
529
542
552
551
540

Fig. 5

# METHOD AND SYSTEM FOR ACCESSING EXTERNALLY-DEFINED OBJECTS FROM AN ARRAY-BASED MATHEMATICAL COMPUTING ENVIRONMENT

## TECHNICAL FIELD

The invention relates generally to mathematical computer programs.

## BACKGROUND

A conventional mathematical tool, such as such as MATLAB™ from MathWorks™, Inc., of Natick, Mass., provides a comprehensive technical computing environment for performing numerical linear algebraic calculations, solving ordinary differential equations, analyzing data, and visualizing solutions to complex mathematical formulas by generating graphs or other images. The computing environment often provides a high-level programming language that includes a variety of operators and programming commands.

Engineers use such mathematical tools for a variety of applications such as designing complex mechanical and electrical control systems, solving optimization problems and performing statistical analysis. In addition, engineers often use mathematical tools in conjunction with a simulation tool for defining and simulating complex mathematical models. For example, manufacturers of mechanical and electronic systems, e.g., cars and integrated circuits, use simulation tools to help them design their products. These tools allow designers to build and test mathematical models of their systems before building a physical prototype. Commercial simulation models can be extremely complex and may include thousands of interconnected functional blocks. Using a simulation tool, a designer can simulate and observe changes in a model over a period of time, typically represented as a series of discrete instants, called time steps, such as 1 millisecond, 1 second, 2 hours, etc. Starting from a set of initial conditions, specified by the designer, the simulation tool drives the model and determines the state of the model at various time steps.

Most technical computing environments provided by conventional mathematical tools are "array-based" such that data types are primarily represented as two-dimensional arrays. In other words, these computing environments do not distinguish between a scalar, a vector, or a matrix. As a result, it is difficult to interface the technical computing environment to an object-oriented environment, such as Java. Because the technical computing environment does not distinguish between scalars, vectors and matrices, it is difficult to invoke methods that have the same name and are only distinguishable by the data types of their input parameters. In addition, it is difficult to translate data from the array-based computing environment of the mathematical tool to the object-oriented environment.

## SUMMARY OF THE INVENTION

In general, the invention provides a method and apparatus, including a computer program apparatus, which facilitate invoking methods of objects defined within an object-oriented environment from a technical computing environment provided by a mathematical tool. In particular, the invention is directed to techniques for invoking methods of objects defined in an object-oriented environment, such as a Java environment, from an array-based computing environment often used in conventional mathematical tools.



When a method is invoked from the computing environment, the techniques automatically compare the input parameters, which are typically arrays, with data types accepted by methods defined within the object-oriented environment. Based on this comparison, the invention automatically selects a method that best accepts the input arrays. The invention, therefore, allows a user to easily invoke methods from external objects, such as Java objects, directly from the technical computing environment of the mathematical tool.

In one aspect, the invention is directed to a technique for invoking a method defined within an object-oriented environment. According to the technique, a list of method signatures corresponding to a particular class and method name is retrieved from the object-oriented environment. Each signature uniquely identifies a corresponding method and lists the method's name and any data types received by the method. After the list is retrieved, the method signatures are ranked by comparing the data types of the signatures with the data types of the input parameters received from the technical computing environment of the mathematical tool. Based on the ranking, one of the method signatures is selected and the corresponding method within the object-oriented environment is invoked unless no suitable method is found, in which case an error condition is raised.

In another aspect, the invention is directed to a computer program, such as a mathematical tool, having instructions suitable for causing a programmable processor to retrieve a list of method signatures from the object-oriented environment. The computer program ranks the method signatures, selects one of the method signatures according to the ranking; and invokes the corresponding method within the object-oriented environment corresponding to the method signature.

In yet another aspect, the invention is directed to a computer system having an object-oriented environment and a mathematical tool executing thereon. The object-oriented environment includes an interface for identifying methods provided by objects defined within the object-oriented environment. The mathematical tool includes a calculation workspace, a command interpreter, and a signature selector. When the command interpreter encounters a reference to a method implemented by an object defined within the object-oriented environment, the command interpreter instructs the signature selector to access the interface of the object-oriented environment to retrieve and rank a list of signatures corresponding to methods defined within the object-oriented environment. The command interpreter invokes one of the methods as a function of the ranking.

The details of various embodiments of the invention are set forth in the accompanying drawings and the description below. Other features and advantages of the invention will become apparent from the description, the drawings, and the claims.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating a system in which a mathematical tool invokes a method of an object defined within an object-oriented computing environment.

FIG. 2 is a flow chart illustrating one embodiment of a process, suitable for implementation in a computer program, in which the mathematical tool invokes the method of the object.

FIG. 3 illustrates one embodiment of a two-dimensional table that stores data types supported by an object-oriented environment ordered by preference..

FIG. 4 illustrates one embodiment of a conversion table suitable for converting data types from an object-oriented environment to an array-based computing environment of a mathematical tool.

FIG. 5 is a block diagram illustrating a programmable processing system suitable for implementing and performing the apparatus and methods of the invention.

## DETAILED DESCRIPTION

FIG. 1 is a block diagram illustrating a system **100** in which mathematical tool **102** invokes an object **110** in an object-oriented environment such as Java environment **120**. Mathematical tool **102** provides a technical computing environment **108** for performing a wide variety of numerical calculations and data analysis operations. Computing environment **108** of mathematical tool **102** is "array-based" such that most data types are represented as arrays of at least two dimensions.

Computing environment **108** of mathematical tool **102** is an interpreted environment that supports a high-level programming language having a variety of operators and programming commands. As the user enters instructions, command interpreter **104** interactively interprets and executes each instruction. Calculation workspace **106** provides a storage area for variables, input data, and resultant data. The user can, for example, define a square matrix within calculation workspace **106** using a single command. The user can directly manipulate the matrix, using one command to find its inverse, another command to find its transpose, or another command to learn its determinant.

In an object-oriented environment, such as Java environment **120**, objects **110** are modules of computer code that specify the data types of a data structure, and also the kinds of operations (or "methods") that can be applied to the data structure. Each object **110** has a corresponding "class" that may be thought of as a prototype that defines the data structures and methods common to all objects of a certain kind. Objects **110** are created at run-time in accordance with their class definition. Thus, each object **110** is a unique instance, referred to as an instantiation, of its corresponding class.

Within a class, each method having the same name must have a different number of inputs, or one or more inputs must differ in data type. Each method has a "signature", which is a unique representation of the method's name and the number and type of each input and output parameter of the method. The method signature is used to distinguish between methods having the same name. For example, in a Java signature, the data types boolean, byte, char, short, int, long, float, and double, are represented in the signature by a single letter: Z, B, C, S, I, J, F, and D, respectively. For all other data types, the signature is an expression of the form "Lclass-name;" where class-name is the name of the corresponding Java class but with dots replaced by the slash character. A void return data type is indicated as a V. Thus, the signature for the method:

```
void sampleMethod(int arg1, double arg2, java-
    .lang.String arg3)
```

has a corresponding signature:

```
(IDLjava/lang/String;)V
```

At the core of Java environment **120** is virtual machine **122**, which provides a self-contained operating environment that is machine independent. Java objects **110** execute within virtual machine **122** regardless of the underlying operating system or hardware and represent any class that virtual machine **122** can see within its scope of execution.

The invention allows a user to easily invoke methods of objects **110** from mathematical tool **102**. This allows the user to exploit the rich functionality offered by Java environment **120**. For example, the user can invoke Java objects **110** in order to quickly design a graphical user interface (GUI). In addition, the user can use Java objects **110**, such as timers and events, within calculation workspace **106**. For example, the user can define and access Java objects **110** from within calculation workspace **106** as follows:

```
jstr=java.lang.String('Hello World');
imFilter.setPixels(5, 5, 100, 100, cm, X, 0, 100);
```

In order for mathematical tool **102** to provide a way for users to invoke objects **110** and their corresponding methods, command interpreter **104** invokes signature selector **112** that automatically determines the appropriate signature of the requested method for invocation. When the user invokes a method provided by one of the objects **110**, command interpreter **105** passes signature selector **112** a name of the method and any input parameters to pass to the method. Because the input parameters are defined in native data types supported by technical computing environment **108**, the parameters are often in the form of an array having any number of dimensions. As described in detail below, signature selector **112** automatically selects a method from object-oriented environment **120** that is best able receive the data from the array inputs.

More specifically, signature selector **112** uses a set of classes within Java environment **120**, referred to herein as selection support classes **124**, to interrogate Java environment **120**. Signature selector **112** passes selection support classes **124** a method name and the name of its corresponding class. Based on the class name and method name, selection support classes **124** determine a set of matching method signatures available within object-oriented environment **120**. In order to communicate with selection support classes **124**, signature selector **112** uses Java native interface (JNI) **126**, which is a programming interface, or API, that allows programs written in C or C++ to invoke Java methods based on a method signature. Signature selector **112** determines and returns the signature of the method available within object-oriented environment **120** that is best able to receive the data from the array inputs. If no suitable methods are found, signature selector **112** returns a null signature. Command interpreter **104** uses selected signature **112** to directly invoke the corresponding object **110** and execute the desired method.

FIG. 2 is a flow chart illustrating one embodiment of a process **200**, suitable for implementation in a computer program application, in which mathematical tool **102** (FIG. 1) invokes a method of a Java object **110** defined within Java environment **120**. When the user seeks to invoke a method provided by one of the Java objects **110**, command interpreter **104** invokes signature selector **112** to automatically determine the appropriate signature of the requested method. Selection support classes **124** interrogate java environment **120** and compiles a list of method signatures having names similar to the requested method and having a matching class name (step **203**).

Next, signature selector **112** calculates a "fitness ranking" for each method signature of the list (step **205**). The fitness ranking indicates how well the input data types of each method match the input parameters passed from calculation workspace **106**, i.e., how well the method is able to receive the data from the input arrays. In order to calculate a signature's fitness ranking, signature selector **112** generates

a "preference value" for each data type specified by the signature by comparing each data type with the input parameters received from workspace **106** (step **207**). For each data type specified by the signature, signature selector **112** references preference table **116**, which maps data types of computing environment **108** to acceptable data types of Java environment **120** ordered by preference.

FIG. **3** illustrates one embodiment of a two-dimensional preference table **116**. Each row of selection preference table **116** corresponds to a unique array type supported by computing environment **108**. For example, row **310** corresponds to input parameters of type array of doubles and lists preferred data types for Java environment **120** as double, float, long, integer, byte and boolean ordered from best fit to worst fit. Thus, for input parameters of type array of doubles, signature selector **112** generates a preference value by determining the location of the corresponding signature data type within row **310**. If the corresponding data type defined by the signature is not found within row **310** then signature selector **112** rejects the signature from the list.

In calculating the preference value for an input data type defined by the signature, signature selector **112** also considers whether the data type of the signature and the corresponding input parameter received from calculation workspace **106** are both classes. If so, signature selector **112** updates the preference value for that signature data type as a function of how many levels separate the two classes within a class hierarchy (step **209**).

Next, signature selector **112** compares the number of dimensions of the input array received from calculation workspace **106** against the number of dimensions of the Java input data type defined by the current signature (step **211**). If the number of dimensions of the input array is larger than the number of dimensions of the Java data type, the signature is rejected because the input array cannot fit into any Java parameter that can be passed to the Java method. If the number of dimensions of the Java data type is larger than that of the input array, the input array is promoted by adding dimensions of length **1**. However, because the match is not perfect, the corresponding preference value is adjusted in proportion to the degree of difference between the number of dimensions of the signature data type and the number of dimensions of the input array. In one implementation, signature selector **112** does not count dimensions of length **1** when determining the number of dimensions. For example, a 5×1 array is considered to have a single dimension.

After calculating a preference value for each data type specified by the signature, signature selector **112** calculates the fitness ranking for the signature according to the individual preference values for the data types defined by the signature (step **213**). It should be noted, however, that signature selector **112** need not explicitly store the calculated preference value for each parameter of the signature. To the contrary, signature selector **112** can calculate the fitness ranking for the signature while iterating over the data types defined by the signature. In one implementation, signature selector **112** initializes a fitness ranking, Fitness_Ranking, to a large number, such as 20, and updates the ranking for each parameter of the current method signature. For example, consider the following method invoked from within workspace **106**:

f=javaObject.example_method(parameter1, parameter2);

Assume parameter1 of the method is a 1×1 array of doubles and parameter2 is a 15×1 array of characters. Consider a method signature defining a first data type long and a second data type array of char having two dimensions. Signature selector **112** subtracts two from Fitness_Ranking because, in selection preference table **116**, the data type long is third of the data types preferred for an input data type array of doubles. Next, signature selector **112** determines that the data type array of char is in the most preferred data type for an input data type of array of characters and, therefore, does not adjust Fitness_Ranking.

Because the parameters are not objects, signature selector **112** does not adjust Fitness_Ranking based on differences in class level. Next, signature selector **112** considers the dimensions and determines that the first data type of the signature, long, is a perfect match dimensionally for a 1×1 array of doubles. Thus, signature selector **112** does not update Fitness_Ranking. However, the two dimensional array of char is one dimension greater than the 15×1 array of characters, so signature selector **112** adjusts Fitness_Ranking by one, resulting in a final value for Fitness_Ranking of **17**.

After calculating fitness rankings for each potential signature, signature selector **112** selects the signature having the highest ranking, unless all of the signatures have been rejected as being unsuitable (step **215**). Signature selector **112** returns the selected signature to command interpreter **104**.

Upon receiving a valid signature, command interpreter **104** invokes the corresponding Java method within object-oriented environment **120** (step **217**). Invoking the Java method has two parts: (1) converting input array parameters from computing environment **108** to input parameters defined by the signature, and (2) converting parameters returned by the method into suitable data types defined within computing environment **108**.

In converting an input array to a data type defined by the signature, argument converter **114** of signature selector **112** generates a Java variable according to the signature and copies data from the input array to newly created variable. Signature selector **112** returns the newly created variable to command interpreter **104** for use when invoking the corresponding method.

If the invoked method has a return value, signature selector **112** examines the signature and determines the dimensions of the return value. Argument converter **114** of signature selector **112** then references conversion table **118** and creates a return variable within workspace **106** for holding the return data. FIG. **4** illustrates one embodiment of a conversion table **118** suitable for converting data types from an object-oriented environment, such as Java environment **120**, to array-based computing environment **108** of mathematical tool **102**. If the return parameter is scalar, then the return variable primarily defaults to a 1×1 array of type double precision floating point. If the Java return value is a rectangular multi-dimensional array, signature selector **112** creates an array having the same number of dimensions as the return value and having the same data type. If, however, the return value is an array of arrays in which the inner arrays have different lengths, then signature selector **112** creates an array of arrays because it cannot create a single, rectangular array. Similarly, signature selector **112** applies this technique for return values of having greater dimensions. After creating the return variable in workspace **106**, signature selector **112** copies data from the return parameters directly into the return variable and passes the return variable to command interpreter **104**.

Various embodiments have been described of a method and system that facilitates invoking methods of objects defined within an object-oriented environment from an array-based technical computing environment often used in conventional mathematical tools. The invention can be

implemented in digital electronic circuitry, or in computer hardware, firmware, software, or in combinations of them. Apparatus of the invention can be implemented in a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor; and method steps of the invention can be performed by a programmable processor executing a program of instructions to perform functions of the invention by operating on input data and generating output. The invention can be implemented advantageously in one or more computer programs that are executable within an operating environment of a programmable system including at least one programmable processor (computer) coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device.

An example of one such type of computer is shown in FIG. 5, which shows a block diagram of a programmable processing system (system) **500** suitable for implementing or performing the apparatus or methods of the invention. As shown in FIG. 5, the system **500** includes a processor **512** that in one embodiment belongs to the PENTIUM® family of microprocessors manufactured by the Intel Corporation of Santa Clara, Calif. However, it should be understood that the invention can be implemented on computers based upon other microprocessors, such as the MIPS® family of microprocessors from the Silicon Graphics Corporation, the POWERPC® family of microprocessors from both the Motorola Corporation and the IBM Corporation, the PRECISION ARCHITECTURE® family of microprocessors from the Hewlett-Packard Company, the SPARC® family of microprocessors from the Sun Microsystems Corporation, or the ALPHA® family of microprocessors from the Compaq Computer Corporation. System **500** represents any server, personal computer, laptop or even a battery-powered, pocket-sized, mobile computer known as a hand-held PC or personal digital assistant (PDA).

System **500** includes system memory **513** (including read only memory (ROM) **514** and random access memory (RAM) **515**, which is connected to the processor **512** by a system data/address bus **516**. ROM **514** represents any device that is primarily read-only including electrically erasable programmable read-only memory (EEPROM), flash memory, etc. RAM **515** represents any random access memory such as Synchronous Dynamic Random Access Memory.

Within the system **500**, input/output bus **518** is connected to the data/address bus **516** via bus controller **519**. In one embodiment, input/output bus **518** is implemented as a standard Peripheral Component Interconnect (PCI) bus. The bus controller **519** examines all signals from the processor **512** to route the signals to the appropriate bus. Signals between the processor **512** and the system memory **513** are merely passed through the bus controller **519**. However, signals from the processor **512** intended for devices other than system memory **513** are routed onto the input/output bus **518**.

Various devices are connected to the input/output bus **518** including hard disk drive **520**, floppy drive **521** that is used to read floppy disk **551**, and optical drive **522**, such as a CD-ROM drive that is used to read an optical disk **552**. The video display **524** or other kind of display device is connected to the input/output bus **518** via a video adapter **525**. Users enter commands and information into the system **500** by using a keyboard **540** and/or pointing device, such as a mouse **542**, which are connected to bus **518** via input/output ports **528**. Other types of pointing devices (not shown in FIG. 5) include track pads, track balls, joysticks, data gloves, head trackers, and other devices suitable for positioning a cursor on the video display **524**.

As shown in FIG. 5, the system **500** also includes a modem **529**. Although illustrated in FIG. 5 as external to the system **500**, those of ordinary skill in the art will quickly recognize that the modem **529** may also be internal to the system **500**. The modem **529** is typically used to communicate over wide area networks (not shown), such as the global Internet. Modem **529** may be connected to a network using either a wired or wireless connection. System **500** is coupled to remote computer **549** via local area network **550**.

Software applications **536** and data are typically stored via one of the memory storage devices, which may include the hard disk **520**, floppy disk **551**, CD-ROM **552** and are copied to RAM **515** for execution. In one embodiment, however, software applications **536** are stored in ROM **514** and are copied to RAM **515** for execution or are executed directly from ROM **514**.

In general, the operating system **535** executes software applications **536** and carries out instructions issued by the user. For example, when the user wants to load a software application **536**, the operating system **535** interprets the instruction and causes the processor **512** to load software application **536** into RAM **515** from either the hard disk **520** or the optical disk **552**. Once one of the software applications **536** is loaded into the RAM **515**, it can be used by the processor **512**. In case of large software applications **536**, processor **512** loads various portions of program modules into RAM **515** as needed.

The Basic Input/Output System (BIOS) **517** for the system **500** is stored in ROM **514** and is loaded into RAM **515** upon booting. Those skilled in the art will recognize that the BIOS **517** is a set of basic executable routines that have conventionally helped to transfer information between the computing resources within the system **500**. Operating system **535** or other software applications **536** use these low-level service routines. In one embodiment system **500** includes a registry (not shown) that is a system database that holds configuration information for system **500**. For example, the Windows® operating system by Microsoft Corporation of Redmond, Wash., maintains the registry in two hidden files, called USER.DAT and SYSTEM.DAT, located on a permanent storage device such as an internal disk.

The invention has been described in terms of particular embodiments. Other embodiments are within the scope of the following claims. For example, the steps of the invention can be performed in a different order and still achieve desirable results. This application is intended to cover any adaptation or variation of the present invention. It is intended that this invention be limited only by the claims and equivalents thereof.

What is claimed is:

**1**. A method for invoking a method defined with an object-oriented computing environment comprising:

retrieving a set of method signatures for a method referenced in a requested method invocation, where each method signature corresponds to a method provided by an object within an object-oriented environment, and further wherein each signature includes a method name and lists any data types of input parameters to be received by the corresponding method;

comparing the data types of input parameters of each method represented by the signatures to data types of input parameters passed by the requested method invo-

cation to determine suitability of each method to receive input parameters passed by the requested method invocation:

ranking the method signatures based on the determined suitability of each method represented by the signatures to receive the input parameters passed by the requested method invocation:

selecting one of the method signatures according to the ranking; and

invoking, in response to the requested method invocation, the method of the object-oriented computing environment corresponding to the selected method signature; wherein the request method invocation is requested by an array-based computing environment provided by a mathematical tool.

2. The method of claim 1, wherein ranking the method signatures comprises calculating a fitness ranking for each signature, the fitness ranking representative of a level of suitability of the data types of the input parameters of the method represented by the signature to use the input parameters passed by the requested method invocation.

3. The method of claim 2, wherein calculating a fitness ranking for each signature includes generating a preference value for each data type of the signature and adjusting the fitness ranking of the corresponding signature as a function of the comparison.

4. The method of claim 2, wherein calculating a fitness ranking for each signature includes calculating a difference in a number of dimensions between the signature data type and the input parameter received from the computing environment.

5. The method of claim 4, wherein the data structure is a two-dimensional array storing, along a first dimension, data types supported by the object-oriented environment ranked according to preference, and further wherein a second dimension of the array corresponds to data types supported by the array-based computing environment.

6. The method of claim 5, wherein the virtual machine is a Java virtual machine.

7. The computer program of claim 6, wherein the computer program invokes the target method by converting the input parameters to data types supported by the object-oriented environment and converting return values from the method to data types supported by the computing environment.

8. The computer program of claim 6, wherein the computer program invokes the method by interpreting the target method with a virtual machine.

9. The computer program of claim 8, wherein the virtual machine is a Java virtual machine.

10. The method of claim 1, wherein, for the signature data types that are superclasses of the data types of the input parameters received from the computing environment, calculating the fitness ranking for each signature includes calculating a difference in level within a class hierarchy for the signature data type and the data type of the corresponding input parameter received from the computing environment.

11. The method of claim 1, wherein comparing each data type of the signature to the data type of the corresponding input parameter includes accessing a data structure storing data types of the object-oriented environment ordered by preference.

12. The method of claim 1, wherein invoking the method includes:

converting the input parameters to data types supported by the object-oriented environment; and

converting return values from the method to data types supported by the computing environment.

13. The method of claim 1, wherein the object-oriented environment includes a virtual machine, and further wherein invoking the method includes interpreting the method via the virtual machine.

14. The method of claim 1, wherein each signature includes a method name comprising the name of the method in the requested method invocation, and wherein each method represented by the signature corresponds to a method provided by the same object.

15. A computer program, tangibly stored on a computer-readable medium, for invoking a method defined within an object-oriented environment, the computer program comprising instructions operable to cause a programmable processor to:

retrieve a set of method signatures for a method referenced in a requested method invocation, where each method signature corresponds to a method provided by an object within an object-oriented environment, and further wherein each signature includes a method name and a data type for each input parameter received by the corresponding method;

compare the data types of each input parameter of each method represented by the signatures to data types of input parameters passed by the requested method invocation to determine suitability of each method to receive the input parameters passed by the requested method invocation;

rank the method signatures based on the determined suitability of each method represented by the signatures to receive the input parameters passed by the requested method invocation;

select one of the method signatures according to the ranking; and

invoke, in response to the requested method invocation, the method of the object-oriented computing environment corresponding to the selected method signature; wherein the request method invocation is requested by an array-based computing environment provided by a mathematical tool.

16. The computer program of claim 15, wherein the computer program ranks the method signatures by calculating a fitness ranking for each signature, the fitness ranking representative of a level of suitability of the data types of the input parameters of the method represented by the signature to use the input parameters passed by the requested method invocation.

17. The computer program of claim 16, wherein the computer program calculates a fitness ranking for each signature by generating a preference value for each data type of the signature and adjusting the fitness ranking of the corresponding signature as a function of the comparison.

18. The computer program of claim 15, for the signature data types that are superclasses of the data types of the input parameters received from the computing environment, the computer program calculates the fitness ranking for each signature by calculating a difference in level within a class hierarchy for the signature data type and the data type of the corresponding input parameter received from the computing environment.

19. The computer program of claim 15, wherein the computer program calculates a fitness-ranking for each signature by calculating a difference in a number of dimensions between the signature data type and the input parameter received from the computing environment.

20. The computer program of claim 15, wherein the computer program compares each data type of the signature to the data type of the corresponding input parameter includes by accessing a data structure storing data types of the object-oriented environment ordered by preference.

21. The computer program of claim 20, wherein the data structure is a two-dimensional array storing, along a first dimension, data types supported by the object-oriented environment ranked according to preference, and further wherein a second dimension corresponds to data types supported by the array-based computing environment.

22. The computer program of claim 15, wherein each signature includes a method name comprising the name of the method in the requested method invocation, and wherein each method represented by the signature corresponds to a method provided by the same object.

23. A system comprising:

an object-oriented environment operating within a computer, wherein the object-oriented environment includes an interface for identifying methods provided by objects within the object-oriented environment; and

a technical computing environment comprising: a calculation workspace; a command interpreter; and

a signature selector, wherein when the command interpreter encounters within the calculation workspace a requested method invocation comprising a reference to a method implemented by an object defined within the object-oriented environment, the command interpreter instructs the signature selector to access the interface of the object-oriented environment to retrieve and rank a list of signatures corresponding to the method referenced in the requested method invocation, wherein the command interpreter ranks the method signatures based on suitability of data types of input parameters of each method represented by the signatures to receive data types of input parameters passed by the requested method invocation and invokes in the object-oriented environment one of the methods represented by one of the signatures selected according to the ranking; wherein the request method invocation is requested by an array-based computing environment provided by a mathematical tool.

24. The system of claim 23, wherein the technical computing environment is provided by a mathematical tool executing on the computer.

25. The system of claim 23, wherein the signature selector ranks the method signatures by calculating a fitness ranking for each signature, the fitness ranking representative of a level of suitability of the data types of the input parameters of the method represented by the signature to use the input parameters passed by the requested method invocation.

26. The system of claim 25, wherein the signature selector calculates a fitness ranking for each signature by:

comparing each data type of an input parameter listed by the signature to a data type of a corresponding input parameter received from the requested method invocation; and

adjusting the fitness ranking of the corresponding signature as a function of the comparison.

27. The system of claim 23, wherein for at least one method signature, the signature selector ranks the method signature by calculating a difference in level within a class hierarchy for the signature data type and the data type of corresponding input parameter received from the computing environment.

28. The system of claim 23, wherein the signature selector determines a preference value for each data type included in the method signatures; and further wherein the computer program calculates the preference value of each signature according to the preference values for the data types included in the signature.

29. The system of claim 23, wherein the signature selector includes a two-dimensional array, wherein along a first dimension the array stores data types supported by the first operating environment ranked according to preference, and further wherein a second dimension corresponds to data types supported by the computing environment.

30. The system of claim 23, wherein the signature selector includes conversion tables to convert the input parameters to data types supported by the object-oriented environment and to convert return values from the method to data types supported by the computing environment.

31. The system of claim 23, wherein the object-oriented environment includes a virtual machine for interpreting the invoked method.

32. The system of claim 31, wherein the virtual machine is a Java virtual machine.

33. The system of claim 23, wherein the interface is a Java Native Interface (JNI).

34. The system of claim 23, wherein for at least one method signature, the signature selector ranks the method signature by calculating a difference in a number of dimensions between the signature data type and the input parameter received from the computing environment.

35. The system of claim 23, wherein each signature includes a method name comprising the name of the method in the requested method invocation, and wherein each method represented by the signature corresponds to a method provided by the same object.

* * * * *