APPENDIX

Nos. 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

APPENDIX

KRISTA S. SCHWARTZ
JONES DAY
77 West Wacker Drive
Chicago, IL 60601-1692
(312) 782-3939

SUSAN M. GERBER
JONES DAY
North Point
901 Lakeside Avenue
Cleveland, OH 44114-1190
(216) 586-3939

GREGORY A. CASTANIAS
JONES DAY
51 Louisiana Avenue, N.W.
Washington, D.C. 20001-2113
(202) 879-3939

Attorneys for Plaintiff-Appellant
The MathWorks, Inc.

WILSON-EPES PRINTING CO., INC. – (202) 789-0096 – WASHINGTON, D. C. 20002

TABLE OF CONTENTS

Page

Description

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.

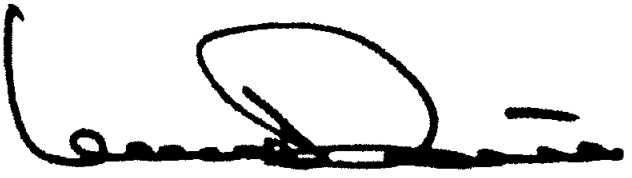
The remainder of the intrinsic evidence supports a construction of "rank[s] [ranking]" that requires an ordinal relationship between the ranked items. The specification broadly discloses a system that ranks method signatures based upon a comparison of the method signatures' data types with the data types 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) rank[s] the method signatures (claim 15, 16) the ranking (claim 1, 15)	placing the method signatures in an ordered manner relative to one another place[s] the method signatures in an ordered manner related to one another 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

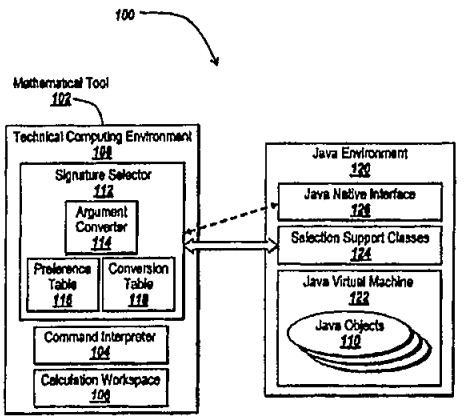
David M. Gay, Symlolic-Algebraic Computations in a Modeling Language for Mathematical Programming, Nov. 1999, Schloss Dagstuhl, p. 4-7.*
Hartmut Pohlheim, Genetic and Evolutionary Algorithm Toolbox for use with MATLAB, Jul. 1997. John W.Eaton, A High-Level Interactive Language for Numeriacal Computaions 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.*
Tiernan, "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

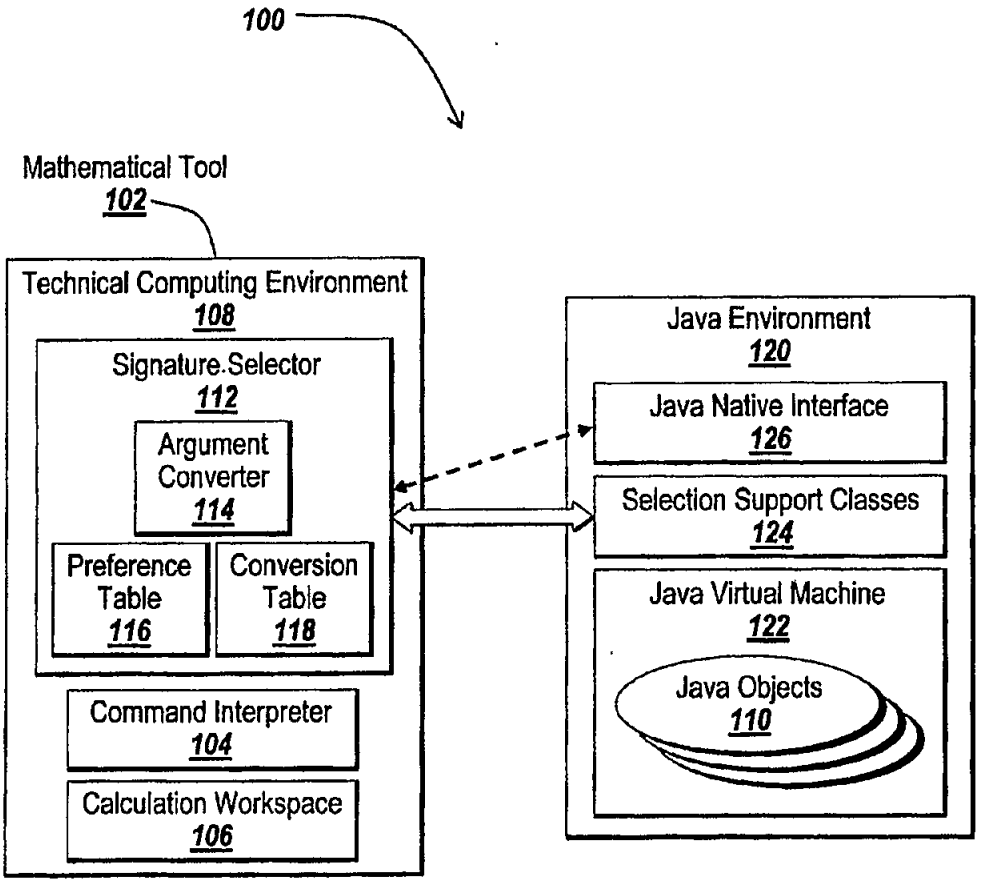

100
Mathematical Tool
102
Technical Computing Environment
108
Signature Selector
112
Argument Converter
114
Preference Table
116
Conversion Table
118
Command Interpreter
104
Calculation Workspace
106
Java Environment
120
Java Native Interface
126
Selection Support Classes
124
Java Virtual Machine
122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

Fig. 1

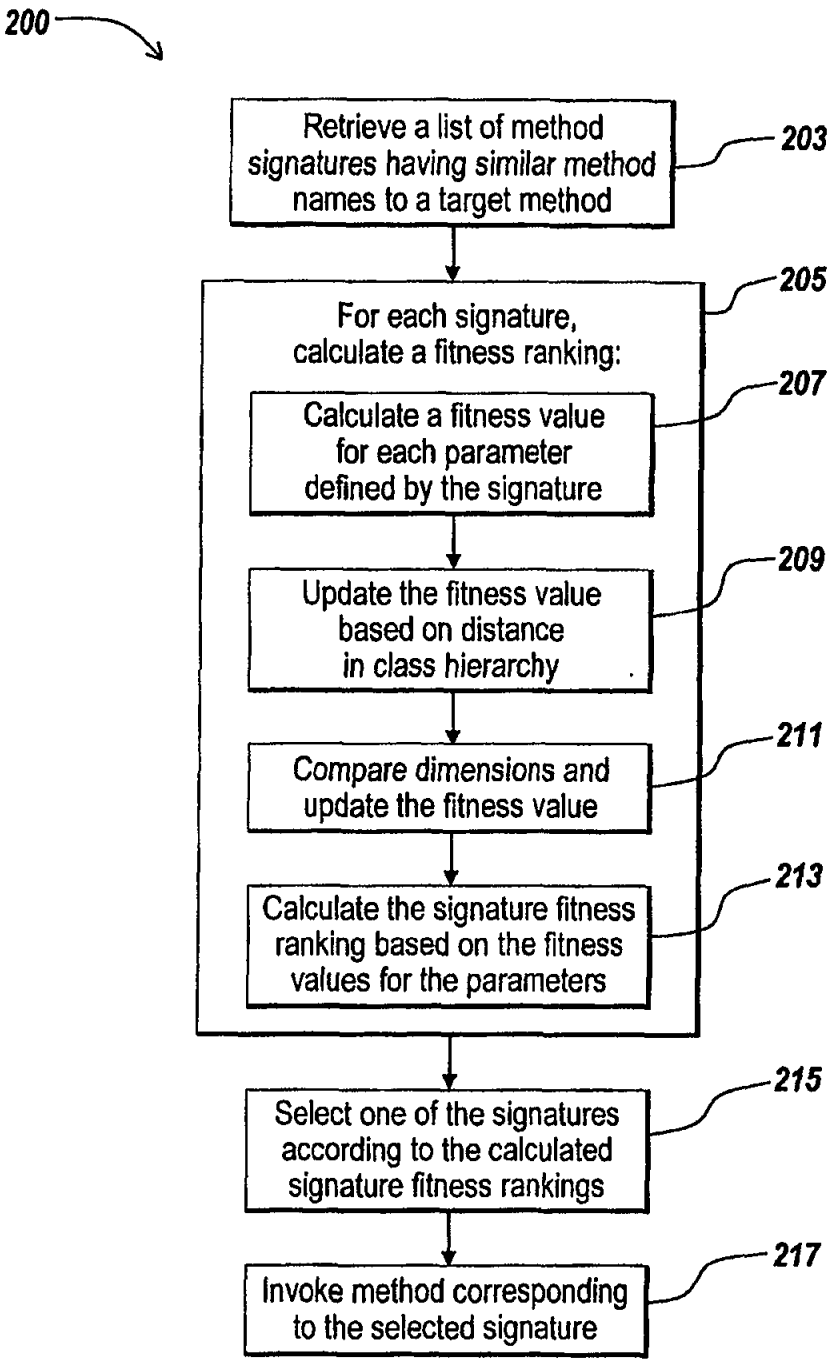

200
Retrieve a list of method signatures having similar method names to a target method
203
For each signature, calculate a fitness ranking:
205
Calculate a fitness value for each parameter defined by the signature
207
Update the fitness value based on distance in class hierarchy
209
Compare dimensions 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

310

Mathematical Tool Data Type	Java Data Type						
	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

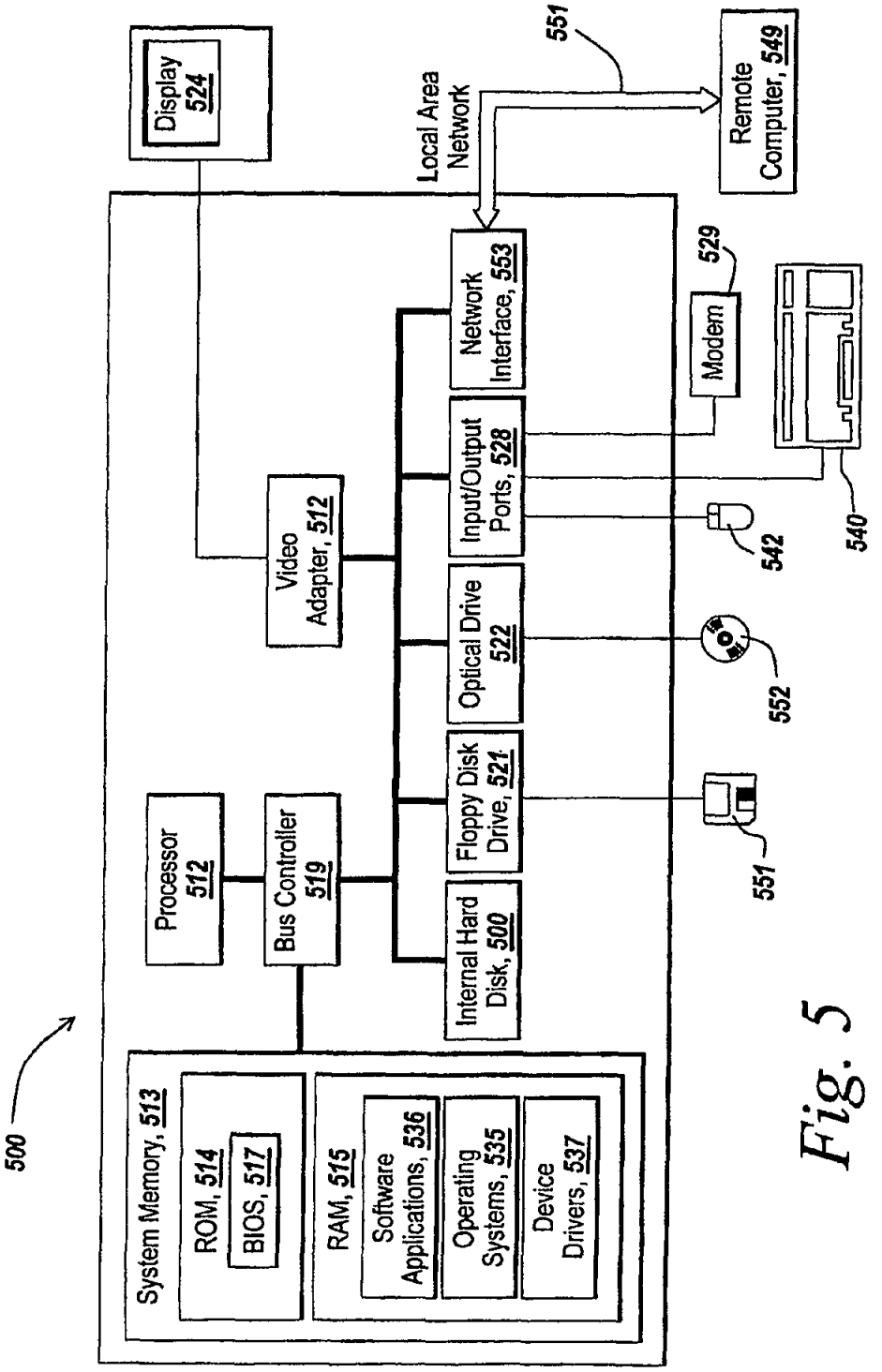

Display 524
Local Area Network
551
Remote Computer, 549
Network Interface, 553
529
Modem
Input/Output Ports, 528
Video Adapter, 512
542
540
Optical Drive 522
552
Floppy Disk Drive, 521
551
Processor 512
Bus Controller 519
Internal Hard Disk, 500
System Memory, 513
ROM, 514
BIOS, 517
RAM, 515
Software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

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');
iml'ilter.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.

* * * * *

Application No.: 09/518287
Art Unit 2126

Docket No.: MWS-064

REMARKS

The Applicants present application contains pending claims 1-34 of which claims 1, 12 and 23 are independent. Claims 1-34 were rejected by the Examiner in the Office Action mailed December 16, 2003. For the reasons set forth below, Applicants respectfully traverse the rejections.

Summary of Claimed Invention

The claimed invention is directed to the calling and use of object methods in an object-oriented environment from a dissimilar array-based (technical) computing environment supplied by a mathematical tool. When a method is called from the array-based computing environment, the data types accepted by the methods in the object-oriented environment are compared with the input parameter data from the array-based computing environment. The methods are ranked based on which methods can best accept the input parameters of the data from the calling array-based computing environment. Based on the comparison, the claimed invention automatically selects a method that best accepts the input arrays.

Rejections under 35 U.S.C. §103(a)

Claims 1, 7, 9, 12, 18 and 23 were rejected under 35 U.S.C. §103(a) as being unpatentable over Cantin et al (EP 0 690 375 A2, hereafter "Cantin et al") in view of a Japanese patent application assigned to NEC Corp., Japanese Application No. 1997JP-0303475("*Index Implementation Method for Object Oriented Database-Involves Comparing Value of Structure Type Member Variable to Obtain Size Related Rank for Variables*", hereafter "NEC"). Claim 1 is an independent method claim upon which claims 7 and 9 are dependent, claim 12 is an independent medium claim upon which claim 18 is dependent, and claim 23 is an independent system claim. For the reasons set forth below, these rejections are respectfully traversed.

Claim Rejections Under 35 U.S.C. §103

I. Claims 1-34 Stand Rejected Under 35 U.S.C. §103

Claims 1-34 stand rejected under 35 U.S.C. §103. Independent claims 1, 12 and 23 stand rejected as being unpatentable over Cantin et al (EP 0 690 375 A2) ("Cantin") in view of Japanese patent application assigned to NEC Corp., Japanese Patent Application No. 1997JP-0303475 ("NEC"). Applicants respectfully traverse this rejection and contend that claims 1-34 are patentable and in condition for allowance.

For ease of discussions, summaries of the claimed invention, the primary reference of Cantin, and secondary reference of NEC are provided below.

A. Summary of Claimed Invention

The claimed invention is directed towards determining what method of an object to call in an object-oriented environment from a technical computing environment by calculating a ranking of method signatures corresponding to the method. A method signature is not an object or a method or a programming instruction, but a character expression representing the specification of the interface of the method. The characters of the method signature's expression describe what data elements the method expects or needs to know about. Each method signature may comprise the method's name and the number and type of each input and output parameter of the method. For example, the java method of "void sampleMethod (int arg1, double arg2, java.lang.string arg3) may have a method signature of "(IDLjava/lang/string;)V". As shown by the example, Java, as with other programming languages, have a vocabulary for specifying method signatures. For example, in the above example and in accordance with Java's vocabulary for method signatures, the letter I in "IDL" of the java method signature for

sampleMethod represents an integer data type, and the letter D a double data type. The method signature itself is not a method that can be executed or invoked, and does not contain any program instruction or computer memory pointer to invoke the method to which it corresponds. It is used to uniquely identify a method and represent the specification of the method's interface by the vocabulary of the method signature language.

Within a class in an object-oriented environment, each method having the same name must have a different number of inputs, or one or more inputs must differ by data type. Since a method signature represents the number and types of inputs to a method, each method having the same name within that class will have a unique method signature. As such, method signatures can be used to uniquely identify the method that should be called when there are multiple methods with the same name that could be called.

In a technical computing environment, most of the data types are represented as arrays of multiple dimensions. Array-based data types do not distinguish between a scalar, vector or a matrix data type. Because the technical computing environment uses array-based data, it is difficult to invoke methods of objects in an object oriented environment that have the same name and are only distinguished by the data types of their input parameters. For example, the technical computing environment may have its own data type that is not available in the object-oriented environment. As such, an input parameter received from a technical computing environment for a method call on an object may not fit agreeably into a data type of the object-oriented environment.

In order to determine an appropriate method to call, the claimed invention compares the technical computing environment data to be provided as input to the method with the specified data types expected as described in the method signature. The method signatures are ranked

based on which corresponding methods are better suited to accept the input parameters of the data from the calling array-based computing environment. Based on the comparison, the claimed invention automatically selects a method signature according to the ranking and then invokes the method corresponding to the selected method signature.

B. Summary of Cantin

The method of Cantin is directed towards having a standard way to implement the mapping of object data to a permanent storage medium in view of the Object Persistence Service Specification (OPSS) as set by the Object Management Group, Inc. Cantin describes the mapping of data contained in an active instance of an object to a permanent storage medium, such as a database. Cantin discusses a specialized implementation of a persistence data service ("PDS") to receive an instance of an object and then store the values set in the properties of the received object to a database table associated with the object. An abstract class is defined in the persistence storage service having custom methods that implement the specific program instructions to store and retrieve the object data to and from an associated database table.

Specifically, Cantin creates a specialized version of an AbstractSchemaMapper base class for each object that is to be stored in a persistent medium. The AbstractSchemaMapper class has four methods to be customized for a given class: SchemaMapperStoreNew, SchemaMapperStoreExisting, SchemaMapperRestore and SchemaMapperDelete. For each of these four methods, custom code is written to map the specific object properties of the object to the specific column names in the database table where the properties will be stored. These methods do not handle persistent storage of any methods of the object.

index key for searching the database. Comparing member variables of a structure type to rank by size is not comparable to comparing the data type of input parameters of a method represented by a method signature to the data type of an input parameter passed by a requested method invocation to determine *suitability of each method to receive input parameters passed by the requested method invocation*. As such, NEC fails to bridge the factual deficiencies of Cantin.

For at least the aforementioned reasons, Applicants submit that Cantin in view of NEC does not detract from the patentability of independent claims 1 and 12. Claims 2-11 depend on and incorporate the patentable subject matter of independent claim 1, and claims 13-22 depend on and incorporate the patentable subject matter of independent claim 12. As such, Applicants submit that Cantin in view of NEC does not detract from the patentability of dependent claims 1 and 12. Accordingly, Applicants respectfully request the withdrawal of the Examiner's rejection of claims 1-22 under 35 U.S.C. §103.

B. Independent Claim 23 Stands Rejected Under 35 U.S.C. §103

Independent claim 23 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Cantin in view of NEC. Applicants respectfully traverse this rejection, and submit that Cantin in view of NEC does not teach or suggest each and every element of independent claim 23, as amended.

Amended claim 23 recites a system comprising an object-oriented environment and a technical computing environment. The object-oriented environment includes an interface for identifying methods provided by objects. The technical computing environment comprises a calculation workspace, a command interpreter, and a signature selector. When the calculation workspace encounters a requested method invocation, the signature selector retrieves and ranks a

list of signatures corresponding to the method referenced in the requested method invocation. The command interpreter invokes in the object-oriented environment one of the methods represented by one of the signatures as a function of the ranking. The ranking *determines 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.* Cantin in view of NEC does not teach or suggest each and every feature recited in amended claim 23.

Cantin does not teach or suggest ranking the signature to *determine 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.* The Examiner cites NEC for the purpose of suggesting one ordinarily skilled in the art might modify Cantin to provide a signature selector to rank a list of signatures according to the claimed invention. NEC discusses ranking the size of variables of data structures stored in the database for using as a composite index key for searching the database. As a database, NEC uses the composite index to search the database to retrieve records from a database. Nowhere does NEC discuss a signature selector retrieving a list of method signatures *and ranking the signatures to determine 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.* As such, NEC fails to bridge the factual deficiencies of Cantin.

For at least the aforementioned reasons, Applicants submit that Cantin in view of NEC does not detract from the patentability of independent claim 23. Claims 24-34 depend on and incorporate the patentable subject matter of independent claim 23. As such, Applicants submit that Cantin in view of NEC does not detract from the patentability of dependent claims 24-34.

Accordingly, Applicants respectfully request the withdrawal of the Examiner's rejection of claims 23-34 under 35 U.S.C. §103.

C. Additional Dependent Claim Rejections Under 35 U.S.C. §103

Dependent claims 3-6, 8, 14-17, 19, 25-29 and 34 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Cantin in view of NEC in further view of Hartmut Pohlheim (*"Genetic and Evolutionary Algorithm Toolbox for use with MATLAB"*).

Dependent claims 2, 13 and 24 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Cantin in view of NEC in further view of Admitted Prior Art.

Dependent claims 10, 11, 20-22, 31 and 32 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Cantin in view of NEC in view of Hartmut Pohlheim and in further view of Bill Venners ("Eternal Math").

Dependent claim 30 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Cantin in view of NEC in further view of John W. Eaton ("A High Level Interactive Language for Numerical Computations, Edition 3 for Octave Version 2.1x").

Dependent claim 33 stands rejected under 35 U.S.C. §103(a) as being unpatentable over Cantin in view of NEC and in further view of David M. Gay ("Symbolic-Algebraic Computations in a Modeling Language for Mathematical Programming").

None of the cited references, alone or in combination, disclose, teach or suggest each and every feature of independent claims 1, 12 and 23, as amended. Claims 2-6, 8, 10 and 11 depend on an incorporate the patentable subject matter of amended independent claim 1. Claims 13-17, 19, and 20-22 depend on an incorporate the patentable subject matter of amended independent claim 12. Claims 24, 30-32 and 34 depend on an incorporate the patentable subject matter of

APPEAL, PATENT

U.S. District Court [LIVE]
Eastern District of TEXAS (Tyler)
CIVIL DOCKET FOR CASE #: 6:06-cv-00334-LED

The MathWorks, Inc. v. COMSOL AB et al
Assigned to: Judge Leonard Davis
Cause: 35:271 Patent Infringement

Date Filed: 07/28/2006
Date Terminated: 03/10/2008
Jury Demand: Both
Nature of Suit: 830 Patent
Jurisdiction: Federal Question

Mediator

Gary V McGowan

Plaintiff

The MathWorks, Inc.

represented by **Mark Nolan Reiter**
Gibson Dunn & Crutcher
2100 McKinney Ave
Suite 1100
Dallas, TX 75201
214-698-3100
Fax: 214 571 2907
Email: mreiter@gibsondunn.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Amanda Aline Abraham
The Roth Law Firm
115 N. Wellington
Suite 200
P.O. Box 876
Marshall, TX 75671-0876
903-935-1665
Fax: 903-935-1797
Email: aa@rothfirm.com
ATTORNEY TO BE NOTICED

Brendan Clay Roth
Law Office of Carl R Roth
115 N Wellington Suite 200
P O Box 876
Marshall, Tx 75670
903/935-1665
Fax: 19039351797
Email: br@rothfirm.com

ATTORNEY TO BE NOTICED

Carl R Roth
The Roth Law Firm, P.C.
115 N Wellington Suite 200
P O Box 876
Marshall, Tx 75670
903/935-1665
Fax: 19039351797
Email: cr@rothfirm.com
ATTORNEY TO BE NOTICED

Christopher Seth Maynard
Jones Day - Dallas
2727 North Harwood Street
Dallas, Tx 75201
214/220-3939
Fax: 214/969-5100
Email: csmaynard@jonesday.com
ATTORNEY TO BE NOTICED

Hilda Contreras Galvan
Jones Day - Dallas
2727 N Harwood St
Dallas, TX 75201
214/969-4556
Fax: 12149695100
Email: hcgalvan@jonesday.com
ATTORNEY TO BE NOTICED

Jason Woodard Cook
Law Office of Jason W Cook
6282 McCommas Blvd
Dallas, TX 75214
214/504-6813
Fax: 469/327-2777
Email: jcook@cookip.com
TERMINATED: 06/13/2007

Krista Sue Schwartz
Jones Day
77 West Wacker Drive
Ste 3500
Chicago, IL 60601
312-782-3939
Fax: 13127828585
Email: ksschwartz@jonesday.com
ATTORNEY TO BE NOTICED

Michael J Newton

Thomas Hart Watkins
Brown McCarroll - Austin
111 Congress Ave
Suite 1400
Austin, TX 78701
512-703-5752
Fax: 512-480-5033
Email: twatkins@mailbmc.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Elizabeth L DeRieux
Capshaw DeRieux, LLP
1127 Judson Road
Ste 220
Longview, TX 75601-5157
903/233-4826
Fax: 903-236-8787
Email: ederieux@capshawlaw.com
ATTORNEY TO BE NOTICED

Gina M McCreadie
Nixon Peabody LLP - Boston
100 Summer Street
Boston, MA 02110
617/345-6189
Fax: 617/345-1300
Email: gmccreadie@nixonpeabody.com

ATTORNEY TO BE NOTICED

Maia H Harris
Nixon Peabody LLP - Boston
100 Summer Street
Boston, MA 02110
617/345-1213
Fax: 617/345-1300
Email: mharris@nixonpeabody.com
ATTORNEY TO BE NOTICED

Marc S Kaufman
Nixon Peabody LLP - Washington DC
401 Ninth Street NW
Suite 900
Washington, DC 20004
202-585-8164
Fax: 202-585-8080
Email: mkaufman@nixonpeabody.com
ATTORNEY TO BE NOTICED

Mark D Robins
Nixon Peabody LLP - Boston
100 Summer Street
Boston, MA 02110
617/345-1000
Fax: 617/345-1300
Email: mrobins@nixonpeabody.com
ATTORNEY TO BE NOTICED

Nicholas G Papastavros
Nixon Peabody LLP - Boston
100 Summer Street
Boston, MA 02110
617/345-1000
Fax: 617/345-1300
Email:
npapastavros@nixonpeabody.com
ATTORNEY TO BE NOTICED

Richard A. Smith
Lynn Tillotson & Pinker LLP
750 N St Paul Street
Suite 1400
Dallas, TX 75201-4639
214/981-3800
Fax: 2149813839
Email: rsmith@lynnllp.com
ATTORNEY TO BE NOTICED

Defendant

COMSOL, Inc. represented by **Michael P Lynn**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Thomas Hart Watkins
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Elizabeth L DeRieux
(See above for address)
ATTORNEY TO BE NOTICED

Gina M McCreadie
(See above for address)
ATTORNEY TO BE NOTICED

Maia H Harris

(See above for address)
ATTORNEY TO BE NOTICED

Marc S Kaufman
(See above for address)
ATTORNEY TO BE NOTICED

Mark D Robins
(See above for address)
ATTORNEY TO BE NOTICED

Nicholas G Papastavros
(See above for address)
ATTORNEY TO BE NOTICED

Richard A. Smith
(See above for address)
ATTORNEY TO BE NOTICED

Counter Claimant

COMSOL AB represented by **Michael P Lynn**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Thomas Hart Watkins
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Gina M McCreadie
(See above for address)
ATTORNEY TO BE NOTICED

Maia H Harris
(See above for address)
ATTORNEY TO BE NOTICED

Mark D Robins
(See above for address)
ATTORNEY TO BE NOTICED

Nicholas G Papastavros
(See above for address)
ATTORNEY TO BE NOTICED

Richard A. Smith
(See above for address)
ATTORNEY TO BE NOTICED

Counter Claimant

COMSOL, Inc. represented by **Michael P Lynn**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Thomas Hart Watkins
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Gina M McCreadie
(See above for address)
ATTORNEY TO BE NOTICED

Maia H Harris
(See above for address)
ATTORNEY TO BE NOTICED

Mark D Robins
(See above for address)
ATTORNEY TO BE NOTICED

Nicholas G Papastavros
(See above for address)
ATTORNEY TO BE NOTICED

Richard A. Smith
(See above for address)
ATTORNEY TO BE NOTICED

V.

Counter Defendant

The MathWorks, Inc. represented by **Mark Nolan Reiter**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Amanda Aline Abraham
(See above for address)
ATTORNEY TO BE NOTICED

Brendan Clay Roth
(See above for address)
ATTORNEY TO BE NOTICED

Carl R Roth
(See above for address)

ATTORNEY TO BE NOTICED

Hilda Contreras Galvan
(See above for address)
ATTORNEY TO BE NOTICED

Jason Woodard Cook
(See above for address)
TERMINATED: 06/13/2007

Krista Sue Schwartz
(See above for address)
ATTORNEY TO BE NOTICED

Michael J Newton
(See above for address)
TERMINATED: 02/28/2007

Terence M Murphy
(See above for address)
ATTORNEY TO BE NOTICED

Todd Richard Miller
(See above for address)
ATTORNEY TO BE NOTICED

Date Filed	#	Docket Text
07/28/2006	1	COMPLAINT against COMSOL AB, COMSOL, Inc. (Filing fee $ 350 pd.) , filed by The MathWorks, Inc. (Attachments: # 1 Exhibit # 2 Civil Cover Sheet)(mjc) (Entered: 07/31/2006)
07/28/2006	2	CORPORATE DISCLOSURE STATEMENT filed by The MathWorks, Inc. identifying no Corporate Parent. (mjc) (Entered: 07/31/2006)
07/28/2006		Filing fee received: $ 350.00, receipt number 6-1-6158 (mjc) (Entered: 07/31/2006)
07/31/2006	3	Form mailed to Commissioner of Patents and Trademarks. (mjc) (Entered: 07/31/2006)
08/09/2006		Summons Issued as to COMSOL AB issued on 8/10/06 and COMSOL, Inc. issued on 8/9/06. (ehs,) (Entered: 08/10/2006)
11/13/2006	4	Return of Service Executed as to COMSOL AB on 11/7/2006, answer due: 11/27/2006. (mjc) (Entered: 11/15/2006)
11/13/2006	5	Return of Service Executed as to COMSOL, Inc. on 11/4/2006, answer due: 11/24/2006. (mjc) (Entered: 11/15/2006)
11/21/2006	6	MOTION for Extension of Time to File Answer re 1 Complaint by COMSOL AB, COMSOL, Inc.. (Lynn, Michael) Additional attachment(s) added on

		11/27/2006 (djh,). (Entered: 11/21/2006)
11/22/2006	7	NOTICE of Disclosure by COMSOL AB, COMSOL, Inc. (Lynn, Michael) (Entered: 11/22/2006)
11/27/2006	8	ORDER granting 6 Motion for Extension of Time to Answer re 6 MOTION for Extension of Time to File Answer re 1 Complaint. Answer due 12/22/06 . Signed by Judge Leonard Davis on 11/27/06. (mjc) (Entered: 11/27/2006)
11/27/2006		Set/Reset Deadlines: COMSOL AB answer due 12/22/2006; COMSOL, Inc. answer due 12/22/2006. (mjc) (Entered: 11/27/2006)
12/22/2006	9	ANSWER to Complaint with Jury Demand by COMSOL AB, COMSOL, Inc..(Smith, Richard) (Entered: 12/22/2006)
01/08/2007	10	NOTICE of Hearing: Scheduling Conference set for 2/26/2007 09:00 AM before Judge Leonard Davis in Tyler. Appendix A is Proposed Discovery Order; Appendix B is Proposed Docket Control Order(mjc) (Entered: 01/08/2007)
01/09/2007	11	NOTICE of Attorney Appearance by Michael J Newton on behalf of The MathWorks, Inc. (Newton, Michael) (Entered: 01/09/2007)
01/09/2007	12	NOTICE of Attorney Appearance by Jason Woodard Cook on behalf of The MathWorks, Inc. (Cook, Jason) (Entered: 01/09/2007)
02/20/2007	13	MOTION for Leave to File *Amended Complaint* by The MathWorks, Inc.. (Attachments: # 1 ***FILED IN ERROR - SEE CORRECTED COMPLAINT 14 ***Proposed Amended Complaint# 2 Exhibit A to Amended Complaint# 3 Exhibit B to Amended Complaint# 4 Text of Proposed Order)(Cook, Jason) Modified on 2/21/2007 (mjc). (Entered: 02/20/2007)
02/21/2007	14	***CORRECTED PROPOSED AMENDED COMPLAINT - REPLACES ATTACHMENT 1 IN 13 *** Additional Attachments to Main Document: 13 MOTION for Leave to File *Amended Complaint*.. (Cook, Jason) Modified on 2/21/2007 (mjc). (Entered: 02/21/2007)
02/21/2007	15	NOTICE by The MathWorks, Inc., COMSOL AB, COMSOL, Inc. *of Parties' Positions Regarding Discovery Order and Docket Control Order* (Attachments: # 1 Draft Agreed Discovery Order# 2 Draft Agreed Docket Control Order)(Cook, Jason) (Entered: 02/21/2007)
02/26/2007	16	Minute Entry for proceedings held before Judge Leonard Davis : Scheduling Conference and Motion Hearing held on 2/26/2007 re 13 MOTION for Leave to File *Amended Complaint* filed by The MathWorks, Inc. (Court Reporter Shea Sloan.) (rlf,) (Entered: 02/26/2007)
02/26/2007	17	ORDER granting 13 Motion for Leave to File Amended Complaint. Ordered that The MathWorks, Inc. re-file the Amended Complaint. Defendants have 10 days to respond. Signed by Judge Leonard Davis on 2/26/07. (mjc) (Entered: 02/26/2007)

02/26/2007	18	AMENDED COMPLAINT against all defendants, filed by The MathWorks, Inc..(Cook, Jason) (Entered: 02/26/2007)
02/26/2007	24	APPLICATION to Appear Pro Hac Vice by Attorney Nicholas G Papastavros for COMSOL AB and COMSOL, Inc. Fee paid, receipt 6-1-8844. Approved 3/1/07. (mjc) (Entered: 03/01/2007)
02/27/2007	19	MOTION to Withdraw as Attorney by The MathWorks, Inc.. (Attachments: # 1 Text of Proposed Order)(Newton, Michael) (Entered: 02/27/2007)
02/28/2007	20	NOTICE of Attorney Appearance by Todd Richard Miller on behalf of The MathWorks, Inc. (Miller, Todd) (Entered: 02/28/2007)
02/28/2007	21	ORDER granting 19 Motion to Withdraw as Attorney. Attorney Michael J Newton terminated . Signed by Judge Leonard Davis on 2/28/07. (mjc) (Entered: 02/28/2007)
02/28/2007	22	ORDER REFERRING CASE to Mediator. Gary V McGowan added as Mediator. Mediation completion deadline is 8/1/07. Signed by Judge Leonard Davis on 2/28/07. (mjc) (Entered: 02/28/2007)
02/28/2007		Set/Reset Scheduling Order Deadlines: Mediation Completion due by 8/1/2007. (mjc) (Entered: 02/28/2007)
03/01/2007	23	APPLICATION to Appear Pro Hac Vice by Attorney Mark D Robins for COMSOL AB and COMSOL, Inc. Fee paid, receipt 6-1-8844. Approved 3/1/07. (mjc) (Entered: 03/01/2007)
03/02/2007	25	NOTICE by The MathWorks, Inc. *of Filing Agreed Proposed Scheduling Order & Discovery Order* (Attachments: # 1 Text of Proposed Order Docket Control Order# 2 Text of Proposed Order Discovery Order)(Cook, Jason) (Entered: 03/02/2007)
03/06/2007	26	SCHEDULING ORDER: Final Pretrial Conference set for 9/18/2008 09:00 AM before Judge Leonard Davis. Discovery due by 4/9/2008. Jury instructions due by 7/23/2008. Jury Selection set for 10/6/2008 09:00AM before Judge Leonard Davis. Mediation Completion due by 8/1/2007. Proposed Pretrial Order due by 7/23/2008. Signed by Judge Leonard Davis on 3/5/07. (mjc) (Entered: 03/06/2007)
03/06/2007		Set/Reset Hearings: Jury Trial set for 10/13/2008 09:00 AM before Judge Leonard Davis. Markman Hearing set for 2/7/2008 09:30 AM before Judge Leonard Davis. (mjc) (Entered: 03/06/2007)
03/06/2007	27	DISCOVERY ORDER entered in furtherance of the management of the Court's docket under FRCP 16. Signed by Judge Leonard Davis on 3/5/07. (mjc) (Entered: 03/06/2007)
03/06/2007	28	Form mailed to Commissioner of Patents and Trademarks. (rml,) (Entered: 03/07/2007)
03/08/2007	29	ANSWER to Amended Complaint *(Patent Infringement)* by COMSOL AB, COMSOL, Inc..(Smith, Richard) (Entered: 03/08/2007)
03/12/2007	30	NOTICE of Disclosure by The MathWorks, Inc. (Cook, Jason) (Entered:

		03/12/2007)
03/29/2007	31	NOTICE of Disclosure by The MathWorks, Inc. (Schwartz, Krista) (Entered: 03/29/2007)
03/29/2007	32	NOTICE of Disclosure by COMSOL AB, COMSOL, Inc. (Robins, Mark) (Entered: 03/29/2007)
04/10/2007	33	Joint MOTION for Extension of Time to File *[Serve] Additional Disclosures* by The MathWorks, Inc., COMSOL AB, COMSOL, Inc.. (Attachments: # 1 Text of Proposed Order)(Cook, Jason) (Entered: 04/10/2007)
04/10/2007	35	APPLICATION to Appear Pro Hac Vice by Attorney Gina M McCreadie for COMSOL AB and COMSOL, Inc. Fee pd., 6-1-9392. Approved 4/11/07 (mjc) (Entered: 04/11/2007)
04/10/2007	36	APPLICATION to Appear Pro Hac Vice by Attorney Maia H Harris for COMSOL AB and COMSOL, Inc. Fee pd., 6-1-9392. Approved 4/11/07. (mjc) (Entered: 04/11/2007)
04/11/2007	34	ORDER granting 33 Joint Motion for Extension of Time to Serve Additional Disclosures. Deadline extended to 5/3/07. Signed by Judge Leonard Davis on 4/11/07. (mjc) (Entered: 04/11/2007)
04/16/2007	37	MOTION Extend Deadline for Invalidity Disclosures with Respect to '745 Patent *(Unopposed)* by COMSOL AB, COMSOL, Inc.. (Harris, Maia) (Entered: 04/16/2007)
04/18/2007	38	ORDER granting 37 Unopposed Motion to Extend Deadline for Invalidity Disclosures to 5/10/07 for U.S. Patent No. 7,181,745 . Signed by Judge Leonard Davis on 4/18/07. (mjc) (Entered: 04/18/2007)
04/20/2007	39	MOTION To Extend Deadline for Invalidity Disclosures with Respect to the '338 Patent *(unopposed)* by COMSOL AB, COMSOL, Inc.. (Attachments: # 1 Text of Proposed Order)(Harris, Maia) (Entered: 04/20/2007)
04/23/2007	40	ORDER granting 39 Defendants' Unopposed Motion to Extend Deadline for Invalidity Disclosures. Deadline to serve the disclosures with respect to the '338 Patent is 5/3/07 . Signed by Judge Leonard Davis on 4/23/07. (mjc) (Entered: 04/23/2007)
05/03/2007	41	NOTICE of Disclosure by The MathWorks, Inc. *of Additional Disclosures in Compliance with Paragraph 2 of the Discovery Order* (Cook, Jason) (Entered: 05/03/2007)
05/08/2007	42	MOTION to Extend Deadline for Invalidity Disclosures with Respect to the '745 Patent *(Unopposed)* by COMSOL AB, COMSOL, Inc.. (Attachments: # 1 Text of Proposed Order)(McCreadie, Gina) (Entered: 05/08/2007)
05/09/2007	43	ORDER granting 42 Unopposed Motion to Extend Deadline for Invalidity Disclosures. Deadline is extended to 5/18/07 . Signed by Judge Leonard Davis on 5/9/07. (mjc) (Entered: 05/09/2007)
05/17/2007		TRANSCRIPT of Proceedings held on 2/26/07 before Judge Leonard Davis. Court Reporter: Shea Sloan. Scheduling Conference and Motion Hearing. See

		16 . (22 pages) shea_sloan@txed.uscourts.gov (sms,) (Entered: 05/17/2007)
05/29/2007	45	Joint MOTION for Protective Order by The MathWorks, Inc., COMSOL AB, COMSOL, Inc.. (Attachments: # 1 Text of Proposed Order (Protective Order)) (Cook, Jason) (Entered: 05/29/2007)
05/30/2007	46	PROTECTIVE ORDER entered pursuant to Rule 26(c) of the FRCP to protect confidential information in the case. Expert Confidentiality Agreement is attached as Exhibit A; Confidentiality Agreement is Exhibit B. Signed by Judge Leonard Davis on 5/30/07. (mjc) (Entered: 05/30/2007)
06/11/2007	47	MOTION to Withdraw as Attorney by The MathWorks, Inc.. (Attachments: # 1 Text of Proposed Order)(Cook, Jason) (Entered: 06/11/2007)
06/13/2007	48	ORDER granting 47 Motion to Withdraw as Attorney. Attorney Jason Woodard Cook terminated . Signed by Judge Leonard Davis on 6/13/07. (mjc) (Entered: 06/13/2007)
06/14/2007	49	MOTION to Amend/Correct *Invalidity Contentions Regarding Patents in Suit (Assented-To)* by COMSOL AB, COMSOL, Inc.. (Attachments: # 1 Exhibit 1# 2 Exhibit 2# 3 Text of Proposed Order)(Harris, Maia) (Entered: 06/14/2007)
06/15/2007	50	ORDER granting 49 Motion to Serve Amended Invailidity Contentions of Defendants COMSOL AB and COMSOL, Inc.. Signed by Judge Leonard Davis on 6/15/07. (mjc) (Entered: 06/15/2007)
08/17/2007	51	MOTION Extend Deadline For Parties to Exchange Proposed Terms and Claim Elements for Construction *(Assented-To)* by COMSOL AB, COMSOL, Inc.. (Attachments: # 1 Text of Proposed Order)(Harris, Maia) (Entered: 08/17/2007)
08/22/2007	52	ORDER granting 51 Motion to Extend Deadline to Exchange Proposed Terms and Claim Elements for Construction. Deadline is extended to 9/14/07. Signed by Judge Leonard Davis on 8/22/07. (mjc) (Entered: 08/22/2007)
08/24/2007	53	REPORT of Mediation by Gary V McGowan. Mediation result: impasse(mjc) (Entered: 08/27/2007)
09/06/2007	54	STIPULATION of Dismissal *and [Proposed] Order Dismissing Claims and Defenses Regarding U.S. Patent No. 7,181,745* by The MathWorks, Inc., COMSOL AB, COMSOL, Inc.. (Schwartz, Krista) (Entered: 09/06/2007)
09/06/2007	55	SEALED MOTION - *Assented-to Motion to Serve Amended Patent Rule 3-1 Disclosure of Asserted Claims and Infringement Contentions* by The MathWorks, Inc.. (Attachments: # 1 Exhibit 1# 2 Text of Proposed Order) (Schwartz, Krista) (Entered: 09/06/2007)
09/10/2007	56	ORDER re 54 Stipulation of Dismissal filed by The MathWorks, Inc., COMSOL AB, COMSOL, Inc., dismissing with prejudice Count II of the Amended Complaint against COMSOL for alleged infringement of U.S. Patent No. 7,181,745. COMSOL dismisses with prejudice any defenses in regard to the '745 Patent; and each party is to bear its own costs. Signed by Judge Leonard Davis on 9/10/07. (mjc) (Entered: 09/10/2007)

09/10/2007	57	ORDER granting 55 Sealed Motion, Assented-to Motion to Serve Amended P.R. 3-1 Disclosure . Signed by Judge Leonard Davis on 9/10/07. (mjc) (Entered: 09/10/2007)
10/10/2007	58	MOTION (Unopposed) to Extend Deadline to Exchange Preliminary Claim Constructions and Extrinsic Evidence by COMSOL AB, COMSOL, Inc.. (Attachments: # 1 Text of Proposed Order)(McCreadie, Gina) (Entered: 10/10/2007)
10/11/2007	59	ORDER granting 58 Motion to Extend Deadline to Exchange Preliminary Claim Constructions and Extrinsic Evidence to 11/1/07 . Signed by Judge Leonard Davis on 10/11/07. (mjc) (Entered: 10/11/2007)
10/24/2007	60	NOTICE of Disclosure by The MathWorks, Inc. (Schwartz, Krista) (Entered: 10/24/2007)
10/25/2007	61	NOTICE of Attorney Appearance by Terence M Murphy on behalf of The MathWorks, Inc. (Murphy, Terence) (Entered: 10/25/2007)
11/01/2007	62	NOTICE of Disclosure by COMSOL AB, COMSOL, Inc. (McCreadie, Gina) (Entered: 11/01/2007)
11/02/2007	63	NOTICE of Disclosure by The MathWorks, Inc. (Schwartz, Krista) (Entered: 11/02/2007)
11/02/2007	64	NOTICE of Disclosure by COMSOL AB, COMSOL, Inc. (McCreadie, Gina) (Entered: 11/02/2007)
11/19/2007	65	NOTICE of Disclosure by The MathWorks, Inc. *of Parties' Joint Claim Construction and Prehearing Statement* (Attachments: # 1 Appendix A# 2 Appendix B)(Schwartz, Krista) (Entered: 11/19/2007)
11/21/2007	66	*Amended* ANSWER to Amended Complaint, COUNTERCLAIM against The MathWorks, Inc. by COMSOL AB, COMSOL, Inc..(Harris, Maia) (Entered: 11/21/2007)
11/29/2007	67	NOTICE by The MathWorks, Inc., COMSOL AB, COMSOL, Inc. *of Proposed Technical Advisor* (Attachments: # 1 Exhibit A)(Schwartz, Krista) (Entered: 11/29/2007)
12/03/2007	68	ORDER appointing Michael T. McLemore as technical advisor to the Court, with costs assessed equally between Plaintiff and Defendants and timely paid as billed. Signed by Judge Leonard Davis on 12/3/07. (mjc) (Entered: 12/03/2007)
12/06/2007	69	MOTION for Extension of Time to File Response/Reply as to 66 Answer to Amended Complaint, Counterclaim by The MathWorks, Inc.. (Attachments: # 1 Text of Proposed Order)(Schwartz, Krista) (Entered: 12/06/2007)
12/07/2007	70	ORDER granting 69 Motion for Extension of Time to File Response/Reply re 69 MOTION for Extension of Time to File Response/Reply as to 66 Answer to Amended Complaint, Counterclaim. Responses due by 12/14/2007. Signed by Judge Leonard Davis on 12/7/07. (mjc) (Entered: 12/07/2007)
12/14/2007	71	SEALED MOTION *to Serve Second Amended Patent Rule 3-1 Disclosure of*

		Asserted Claims and Infringement Contentions by The MathWorks, Inc.. (Attachments: # 1 Exhibit A# 2 Text of Proposed Order)(Schwartz, Krista) (Entered: 12/14/2007)
12/14/2007	72	MOTION to Dismiss *Counterclaims and Strike Defendants' Amended Answer and Counterclaim* by The MathWorks, Inc.. (Attachments: # 1 Exhibit A# 2 Exhibit B# 3 Exhibit C# 4 Exhibit D# 5 Text of Proposed Order)(Schwartz, Krista) (Entered: 12/14/2007)
12/17/2007	73	ORDER granting 71 Sealed Motion to Serve Second Amended Patent Rule 3-1 Disclosure of Asserted Claims and Infringement Contentions. Signed by Judge Leonard Davis on 12/17/07. (mjc) (Entered: 12/17/2007)
12/26/2007	74	STIPULATION *and [Proposed] Order Regarding Counterclaims* by The MathWorks, Inc.. (Schwartz, Krista) (Entered: 12/26/2007)
12/28/2007	75	ORDER approving 74 Stipulation and Order Regarding Counterclaims filed by The MathWorks, Inc. The pleadings are hereby amended to reflect the stipulations in this Order. Signed by Judge Leonard Davis on 12/28/07. (mjc) (Entered: 12/28/2007)
12/31/2007	76	NOTICE by The MathWorks, Inc. re 72 MOTION to Dismiss *Counterclaims and Strike Defendants' Amended Answer and Counterclaim : The MathWorks, Inc.'s Notice of Withdrawal of Motion* (Schwartz, Krista) (Entered: 12/31/2007)
12/31/2007	77	ANSWER to Counterclaim by The MathWorks, Inc..(Schwartz, Krista) (Entered: 12/31/2007)
01/03/2008	78	CLAIM CONSTRUCTION BRIEF filed by The MathWorks, Inc.. (Attachments: # 1 Exhibit 1# 2 Exhibit 2# 3 Exhibit 3# 4 Exhibit 4# 5 Appendix Part 1 of 20# 6 Appendix Part 2 of 20# 7 Appendix Part 3 of 20# 8 Appendix Part 4 of 20# 9 Appendix Part 5 of 20# 10 Appendix Part 6 of 20# 11 Appendix Part 7 of 20# 12 Appendix Part 8 of 20# 13 Appendix Part 9 of 20# 14 Appendix Part 10 of 20# 15 Appendix Part 11 of 20# 16 Appendix Part 12 of 20# 17 Appendix Part 13 of 20# 18 Appendix Part 14 of 20# 19 Appendix Part 15 of 20# 20 Appendix Part 16 of 20# 21 Appendix Part 17 of 20# 22 Appendix Part 18 of 20# 23 Appendix Part 19 of 20# 24 Appendix Part 20 of 20)(Schwartz, Krista) (Entered: 01/03/2008)
01/04/2008	79	NOTICE of Attorney Appearance by Christopher Seth Maynard on behalf of The MathWorks, Inc. (Maynard, Christopher) (Entered: 01/04/2008)
01/07/2008		E-NOTICE of Hearing: Jury Trial set for 10/13/2008 @ 9:00 **HAS BEEN CONTINUED and JURY TRIAL RESET to 10/14/2008 @ 09:00 AM** before Judge Leonard Davis. (NOTE: JURY SELECTION OF 10/6/2008 WILL REMAIN THE SAME)(rlf,) (Entered: 01/07/2008)
01/08/2008	80	NOTICE of Designation of Attorney in Charge to Terence M Murphy on behalf of The MathWorks, Inc. (Murphy, Terence) (Entered: 01/08/2008)
01/16/2008	81	NOTICE of Attorney Appearance by Thomas Hart Watkins on behalf of all defendants (Watkins, Thomas) (Entered: 01/16/2008)

01/17/2008		E-NOTICE of Hearing:**Markman Hearing set for 2/7/2008 at 9:30 HAS BEEN CONTINUED AND THE MARKMAN HEARING IS RESET TO 2/6/2008 AT 9:30 AM** before Judge Leonard Davis. (rlf,) (Entered: 01/17/2008)
01/17/2008	82	SEALED PATENT RESPONSE by COMSOL AB, COMSOL, Inc. to 78 Claim Construction Brief,, *filed by COMSOL AB, COMSOL, Inc.*. (Attachments: # 1 Exhibit 1# 2 Exhibit 2# 3 Exhibit 3# 4 Exhibit 4# 5 Exhibit 5# 6 Exhibit 6# 7 Exhibit 7)(McCreadie, Gina) (Entered: 01/17/2008)
01/24/2008	83	NOTICE by The MathWorks, Inc., COMSOL AB, COMSOL, Inc. *Stating Estimated Time Requested for February 6 Hearing* (Schwartz, Krista) (Entered: 01/24/2008)
01/24/2008	84	CLAIM CONSTRUCTION REPLY BRIEF filed by The MathWorks, Inc. re 82 Sealed Patent Response to Non-Motion,. (Attachments: # 1 Exhibit 5# 2 Exhibit 6# 3 Exhibit 7)(Schwartz, Krista) (Entered: 01/24/2008)
01/31/2008	85	NOTICE of Attorney Appearance by Elizabeth L DeRieux on behalf of COMSOL AB, COMSOL, Inc. (DeRieux, Elizabeth) (Entered: 01/31/2008)
02/05/2008	86	***FILED IN ERROR. PLEASE DISREGARD.*** MOTION for Leave to Appear Pro Hac Vice *of Marc Kaufman* by COMSOL AB, COMSOL, Inc.. (Watkins, Thomas) Modified on 2/6/2008 (mjc). (Entered: 02/05/2008)
02/05/2008	87	NOTICE of Attorney Appearance by Susan M Gerber on behalf of The MathWorks, Inc. (Gerber, Susan) (Entered: 02/05/2008)
02/06/2008	88	Minute Entry for proceedings held before Judge Leonard Davis : Markman Hearing held on 2/6/2008. (Court Reporter Shea Sloan.) (rlf,) (Entered: 02/06/2008)
02/06/2008	89	APPLICATION to Appear Pro Hac Vice by Attorney Marc S Kaufman for COMSOL AB and COMSOL, Inc. Fee pd., 6-1-12691. Approved 2/6/08. (mjc) (Entered: 02/07/2008)
02/11/2008	90	ORDER to promptly pay Technical Consultant Michael McLemore $20,000 for services rendered through 2/6/2008, with plaintiff and defendants each assessed $10,000 of the cost. Signed by Judge Leonard Davis on 2/11/08. (mjc) (Entered: 02/11/2008)
02/13/2008	91	MEMORANDUM OPINION construing the terms in U.S. Patent No. 7,051,338. 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. Signed by Judge Leonard Davis on 2/12/08. (mjc) (Entered: 02/13/2008)
02/21/2008	92	NOTICE of Attorney Appearance by Brendan Clay Roth on behalf of The MathWorks, Inc. (Roth, Brendan) (Entered: 02/21/2008)
02/21/2008	93	NOTICE of Attorney Appearance by Amanda Aline Abraham on behalf of The MathWorks, Inc. (Abraham, Amanda) (Entered: 02/21/2008)

02/27/2008	94	MOTION for Extension of Time to File *Expert Witness Designations and Exchange Expert Reports* by The MathWorks, Inc., COMSOL AB, COMSOL, Inc.. (Attachments: # 1 Text of Proposed Order)(Schwartz, Krista) (Entered: 02/27/2008)
02/28/2008	95	ORDER granting 94 Motion for Extension of Time to Designate Expert Witnesses and Exchange Expert Witness Reports. Signed by Judge Leonard Davis on 2/28/08. (mjc) (Entered: 02/28/2008)
02/28/2008	96	NOTICE by COMSOL AB, COMSOL, Inc. re 90 Order of payment to technical advisor (mjc) (Entered: 02/29/2008)
03/07/2008	97	STIPULATION by The MathWorks, Inc., COMSOL AB, COMSOL, Inc.. (Attachments: # 1 Exhibit A (Proposed Final Judgment))(Schwartz, Krista) (Entered: 03/07/2008)
03/10/2008	98	FINAL JUDGMENT re Stipulation of the parties: COMSOL admits infringing claims 1-4, 11-17, 19-20 and 22 of U.S. Patent No. 7,051,338, but not other claims. The Court enjoins COMSOL and its representatives, for the life of the above listed claims, from manufacturing, using, selling, promoting, distributing, displaying or otherwise disposing of in any manner in the U.S. COMSOL Script 1.0, 1.0a and 1.0b. COMSOL Counterclaim Count I for Declaratory Judgment of Non-Infringement is dismissed 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. COMSOL Counterclaim Count II for Declaratory Judgment of Invalidity is dismissed without prejudice. Each party to bear its own costs. Signed by Judge Leonard Davis on 3/10/08. (mjc) (Entered: 03/10/2008)
03/24/2008	99	NOTICE OF APPEAL - PATENT CASE as to 98 Judgment,,, by The MathWorks, Inc.. Filing fee $ 455, receipt number 1474300. (Schwartz, Krista) (Entered: 03/24/2008)
03/26/2008	100	Forwarded Notice of Appeal on a Patent Case to The U S Court of Appeals for the Federal Circuit with certified docket sheet and certified final judgment (djh,) (Entered: 03/26/2008)
04/03/2008	101	TRANSCRIPT REQUEST by The MathWorks, Inc. re 99 Notice of Appeal - PATENT CASE (Schwartz, Krista) (Entered: 04/03/2008)
04/08/2008		TRANSCRIPT of Markman Proceedings held on 2/6/08 before Judge Leonard Davis. Court Reporter: Shea Sloan. 29 pages. See 104 . shea_sloan@txed.uscourts.gov.(sms,) (Entered: 04/08/2008)
04/11/2008	103	U S Court of Appeals for the Federal Circuit acknowledgment of receipt of Notice of Appeal and assigned case number 2008-1283 (djh,) (Entered: 04/11/2008)

PACER Service Center
Transaction Receipt
05/19/2008 15:13:38

UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF TEXAS

THE MATHWORKS, INC.,

Plaintiff,

v.

No. 6:06 Civ. 334 (LED)

COMSOL AB and COMSOL, INC.,

Defendants.

AMENDED ANSWER OF DEFENDANTS COMSOL AB AND COMSOL, INC. TO AMENDED PATENT INFRINGEMENT COMPLAINT AND COUNTERCLAIM

Pursuant to Rule 8 of the Federal Rules of Civil Procedure, Defendants COMSOL AB and COMSOL, Inc. (collectively "COMSOL") hereby answer the amended patent infringement complaint ("Complaint") of Plaintiff The MathWorks, Inc. ("MathWorks") as follows:

PARTIES

1. COMSOL lacks knowledge or information sufficient to form a belief as to the truth of the allegations contained in Paragraph 1, and, therefore, denies such allegations.

2. COMSOL denies that COMSOL AB has offices at Bjornnasvagen 21, S113 47, Stockholm, Sweden. COMSOL admits the other allegations contained in Paragraph 2 of the Complaint.

3. COMSOL admits the allegations contained in Paragraph 3 of the Complaint.

JURISDICTION AND VENUE

4. Paragraph 4 contains legal assertions to which no response is required. To the extent that the allegations contained in Paragraph 4 require a response, they are denied.

5. Paragraph 5 contains legal assertions to which no response is required. To the extent that the allegations contained in Paragraph 5 require a response, they are denied.

6. Paragraph 6 contains legal assertions to which no response is required. To the extent that the allegations contained in Paragraph 6 require a response, they are denied.

COUNT I
Infringement of United States Patent No. 7,051,338

7. COMSOL repeats and incorporates by reference its responses to each of the preceding Paragraphs as though fully set forth herein.

8. COMSOL admits that United States Patent No. 7,051,338, entitled "Method and System for Accessing Externally-Defined Objects From An Array-Based Mathematical Computing Environment," ("the '338 Patent") issued on May 23, 2006. COMSOL denies that the '338 Patent is attached to the Complaint as Exhibit A, and denies the allegation that the '338 Patent was duly and properly issued. COMSOL is without knowledge or information sufficient to form a belief as to the truth of the remaining allegations contained in Paragraph 8, and, therefore, denies such allegations.

9. COMSOL admits that version 1.0b of COMSOL Script™ fell within the scope of one or more claims of the '338 Patent, and that version 1.0b of COMSOL Script™ was manufactured, used, offered for sale and/or sold in the United States from May 23, 2006 through September 13, 2006. COMSOL denies all other allegations contained in Paragraph 9 of the Complaint, and specifically denies that any other product (including but not limited to any other version of COMSOL Script™) has at any point in time fallen within the scope of one or more claims of the '338 Patent.

10. COMSOL denies the allegations contained in Paragraph 10 of the Complaint.

11. COMSOL denies the allegations contained in Paragraph 11 of the Complaint.

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

The MathWorks, Inc., Plaintiff, v. COMSOL AB and COMSOL, Inc., Defendants.	)	Civil Action No. 6:06-cv-334-LED JURY TRIAL DEMANDED

THE MATHWORKS, INC.'S
OPENING CLAIM CONSTRUCTION BRIEF

V. MATHWORKS' PROPOSED CLAIM CONSTRUCTIONS SHOULD BE ADOPTED

Construction of the following five claim terms is in dispute:[5]

Disputed Term or Phrase
fitness ranking / fitness-ranking
rank(s) / ranking
array-based computing environment
mathematical tool
signature(s) / method signatures(s)

A. "Fitness Ranking" / "Fitness-Ranking"

MathWorks' Proposed Construction	COMSOL's Proposed Construction
No construction necessary If the Court believes a construction is warranted: value representing the level of suitability of the data types of a method to receive the input parameters passed by the mathematical tool	A numerical value indicating how well the method is able to receive the input parameters

The language of the '338 patent claims themselves, the starting and ending point for a proper claim construction, clearly defines the term "fitness ranking" / "fitness-ranking." As such, no claim construction is necessary for this term. Each claim that includes the phrase "fitness ranking" / "fitness-ranking" also includes a definition of the phrase in the claim itself. For example, claim 2 recites:

> 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***

[5] A table setting forth all the agreed and disputed constructions as well as identifying the asserted claims in which the terms appear is attached as Exhibit 4.

represented by the signature to use the input parameters passed by the requested method invocation.[6]

'338 patent at col. 9, ll. 16-21 (emphasis added). Because fitness ranking is defined by the claims themselves, no construction is necessary. *See Biotec*, 249 F.3d at 1349; *Collegenet*, 2004 WL 2429843, at *14; *see also Interactive Gift Express*, 256 F.3d at 1331.

The specification of the '338 patent supports the definition of "fitness ranking" provided by the claim language of the '338 patent. The preferred embodiment of the '338 patent attempts to find a method in the object-oriented environment that is the closest fit for the data types passed by the array-based mathematical tool. To do so, the array-based mathematical tool evaluates the data types of each input parameter of a candidate method against the data types to be passed by the array-based mathematical tool and calculates a value to reflect the ***level*** of suitability of the method to receive the passed data types. The specification refers to this value as the "fitness ranking." *See, e.g.*, '338 patent at Fig. 2 and col. 4 at ll. 61-66. Although the fitness ranking of the preferred embodiment is a numerical value, there is nothing in the intrinsic record that requires fitness ranking to be a number. Therefore, COMSOL's proposed construction of "fitness ranking," which would require a numerical value, is improperly limited to the preferred embodiment and does not accord the term the full breadth to which it is entitled. Furthermore, COMSOL's proposal does not track the language of the claim, making no mention of suitability or data types, despite the patentees' decision to include these words in the claim language defining "fitness ranking."

Although MathWorks believes that this term does not require construction because the claim language itself defines "fitness ranking" / "fitness-ranking," if the Court elects to construe

[6] Claim 16 similarly defines fitness ranking: "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." '338 patent at col. 10, ll. 43-49.

this term, a proper construction should track the language of the claims using words that are helpful to the jury and should not be limited to the preferred embodiment. Such a construction is: **value representing the level of suitability of the data types of a method to receive the input parameters passed by the mathematical tool.**

B. "Rank(s)" / "Ranking"

MathWorks' Proposed Construction	COMSOL's Proposed Construction
rank(s) / ranking (when used as a verb): to assign to a particular class rank(s) / ranking (when used as a noun): assignment to a particular class	rank(s) the method signatures / ranking the method signatures: Placing the method signatures in an ordered manner relative to one another rank a list of signatures: Placing a list of 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

MathWorks' proposed construction flows directly from the way in which the patentees used this term in the claims, specification and prosecution history. *See, e.g., Phillips*, 415 F.3d at 1315. MathWorks has adhered to the rule against limiting claims to the preferred embodiment. *See, e.g., Electro Med. Sys.*, 34 F. 3d at 1054. Furthermore, MathWorks' definition accords "ranking" a broader scope than "fitness ranking" as required by the doctrine of claim differentiation. As such, MathWorks' proposed construction is well-supported by the intrinsic record and accords the term the full breadth permitted by the disclosure of the specification. COMSOL, on the other hand, disregards these rules of claim construction and attempts to unduly restrict the definition of "ranking" to one technique used by the preferred embodiment. COMSOL's proposal vitiates the difference in scope between claims using the term "ranking"

UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF TEXAS
TYLER DIVISION

THE MATHWORKS, INC.,	§	
	§	
Plaintiff,	§	Civil Action No. 6:06CV334-LED
	§	
v.	§	FILED UNDER SEAL
	§	CONTAINS DESIGNATED
COMSOL AB and COMSOL, INC.,	§	CONFIDENTIAL INFORMATION
	§	
Defendants.	§	

COMSOL AB AND COMSOL, INC.'S RESPONSE TO THE MATHWORKS, INC.'S OPENING CLAIM CONSTRUCTION BRIEF

CONFIDENTIAL – SUBJECT TO PROTECTIVE ORDER: Exhibit 4 to this brief contains information that is the subject of a Protective Order of this Court and cannot be made available to anyone other than this Court or counsel of record for the parties.

B. "Rank(s) The Method Signatures / Ranking The Method Signatures" Should Be Construed To Mean: "placing the method signatures in an ordered manner relative to one another;" And "The Ranking" Should Be Construed To Mean: "the list of method signatures placed in an ordered manner relative to one another."

Term or Phrase	COMSOL'S Proposed Construction	The MathWorks' Proposed Construction
ranks the method signature/ranking the method signature	placing the method signatures in an ordered manner relative to one another	(with respect to the proposed terms rank(s)/ranking/ranked): No construction necessary. If the Court believes a construction is warranted: when used as a verb: to assign to a particular class when used as a noun: assignment to a particular class
the ranking	the list of method signatures placed in an ordered manner relative to one another	no proposed construction specifically provided; *see* proposed construction of "ranking"

COMSOL's proposed construction of these terms is consistent with the ordinary meaning of the phrase "rank," and captures the full context of how these phrases are used throughout the specification and in the claims of the '338 Patent. When taken together, the intrinsic and extrinsic evidence clearly indicate that "**ranking**" of signatures involves placing the signatures on some sort of ordinal scale, i.e., in order from highest to lowest, and that the resulting "**ranking**" constitutes a list of method signatures placed in an ordered manner relative to one another.

1. Standard English dictionary references support COMSOL's proposed construction.

The '338 Patent does not provide an express definition of the term "rank" and the various phrases containing the term. However, entries in standard English dictionaries for the terms

Volume 1, Pages 1-162

Exhibits: 1-9

UNITED STATES DISTRICT COURT

EASTERN DISTRICT OF TEXAS

TYLER DIVISION

THE MATHWORKS, INC.,

Plaintiff

v. Docket No. 6:06CV334-LED

COMSOL AB and COMSOL, INC.,

Defendants

VIDEOTAPED DEPOSITION OF DAVID A. FOTI

Tuesday, December 11, 2007, 10:20 a.m.

Nixon Peabody LLP

100 Summer Street

Boston, Massachusetts

----Reporter: Kathleen Mullen Silva, RPR, CRR----

ksilva@fabreporters.com www.fabreporters.com

Farmer Arsenault Brock LLC

50 Congress Street, Suite 415

Boston, Massachusetts 02109

617.728.4404 fax 617.728.4403

REDACTED

Q. Anywhere in the patent does the invention show the retrieval of zero signatures?

A. Well, right here it says a list of method signatures; and anyone of normal skill in the art would know that lists in computer science, generally speaking, can be of zero or greater length.

REDACTED

MR. PAPASTAVROS: I'm sorry. I have to take a quick bathroom break.

THE VIDEOGRAPHER: The time is 1:30. We are off record.

(A recess was taken.)

MR. PAPASTAVROS: Mr. Kaufman is going to ask you some questions.

MS. GERBER: No.

MR. PAPASTAVROS: He's going to ask some questions.

The MathWorks, Inc.'s Technology Tutorial

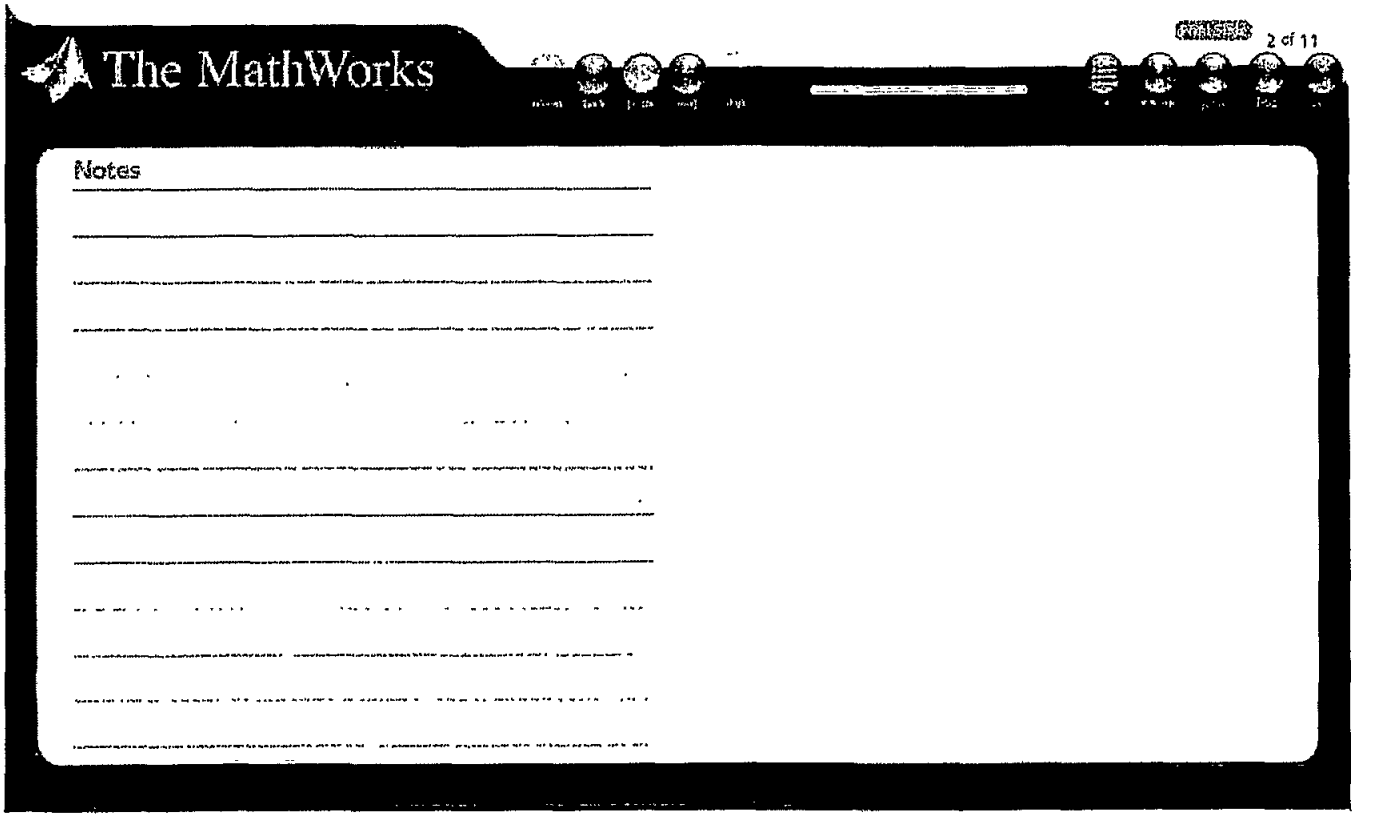
for U.S. Patent No.

7,051,338

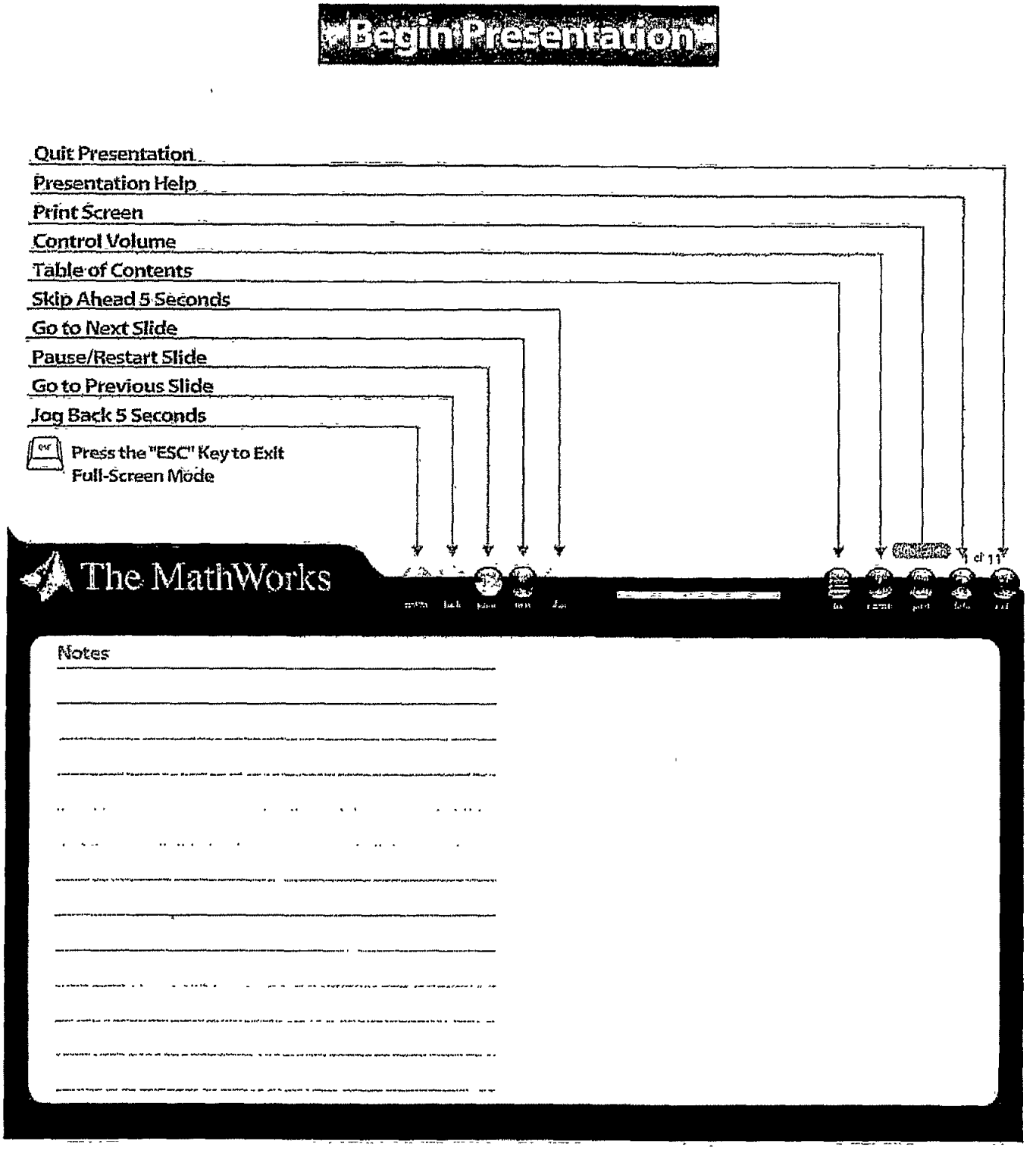
The MathWorks, Inc. v. COMSOL AB, et al.

6:06-CV-334

Before Judge Leonard E. Davis



Operating Instructions



Computer Programming

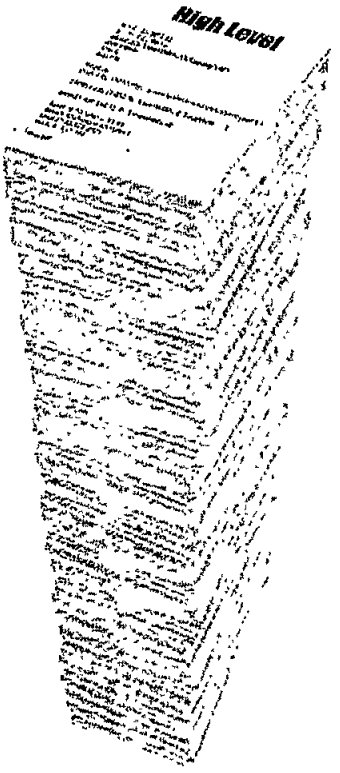
The MathWorks

Notes

Computers operate in their own language– called machine code. It is difficult for human beings to understand machine code. Accordingly, humans have designed ways to program computers at a higher level of abstraction, using words and expressions that are easier for a human being to follow and comprehend.

This higher level of computer code is used by human beings to create computer programs. However, computers cannot understand and execute the higher-level code without an intermediary. Compilers and interpreters are the intermediary that essentially translate the higher level code into the computer's own machine code.

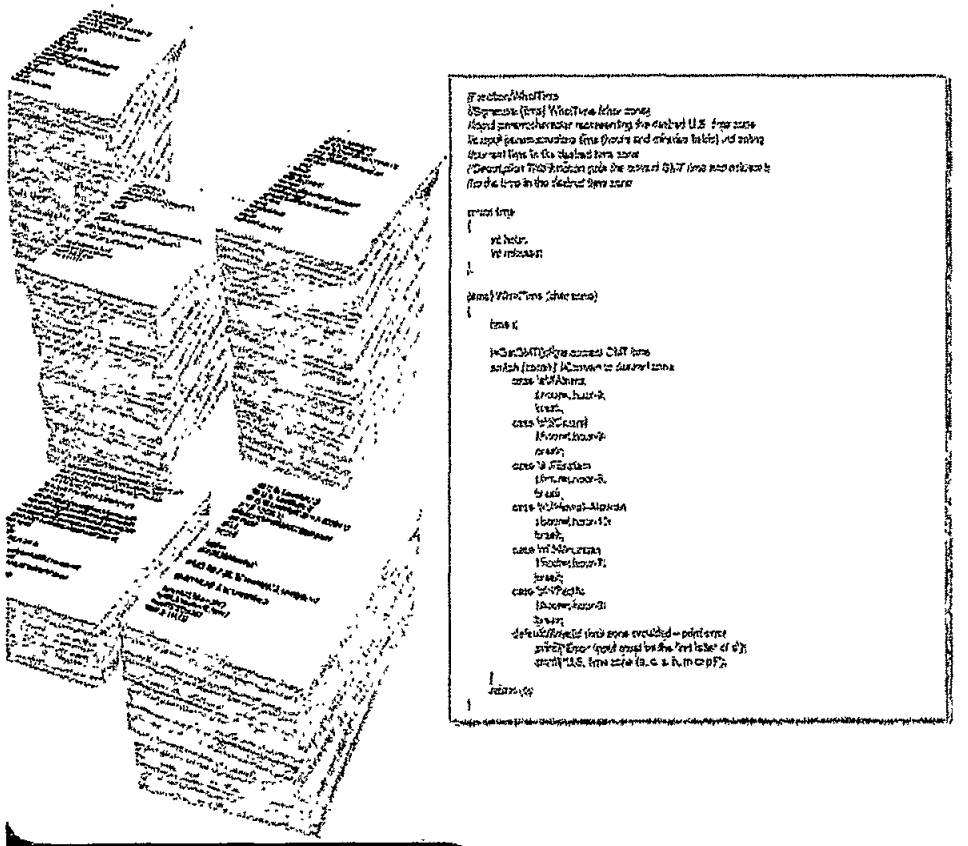
Computer Programming



Notes

Some programs are very large. For example, the MATLAB program includes millions of lines of computer code. A large program like MATLAB must be organized so that the computer code is easier for programmers to understand and manipulate.

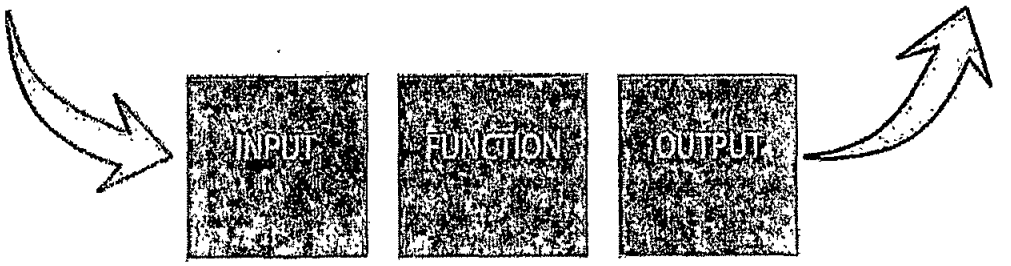
Functional Programming



Notes

Historically, computer programs were organized in a manner that is referred to as functional programming. Functional programming groups lines of code together into functions. Each function has a name that refers to that particular function, or group of code, and the name of the function is essentially a shortcut to refer to the entire group of code that is in that function.

Functional Programming



Notes

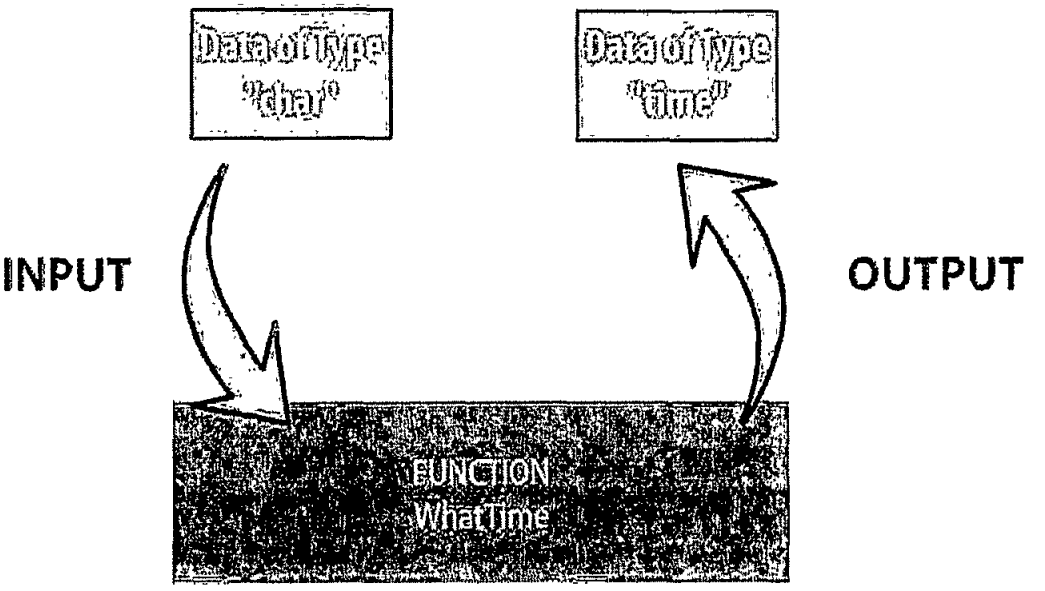
Often, programmers need to move data into and out of functions. Data provided to a function is called an input parameter. Data received from a function is called an output parameter.

Each piece of data provided to or received from a function must be of the data type that the function expects.

Computer programs manipulate data, such as dollar amounts, times, words, or numbers.

All data is not stored in the same manner and cannot be manipulated in the same manner. For example, you can multiply numbers together, but you cannot multiply words together.

Accordingly, it is important for a function to receive the data type that it is expecting so it will be able to manipulate the data appropriately.

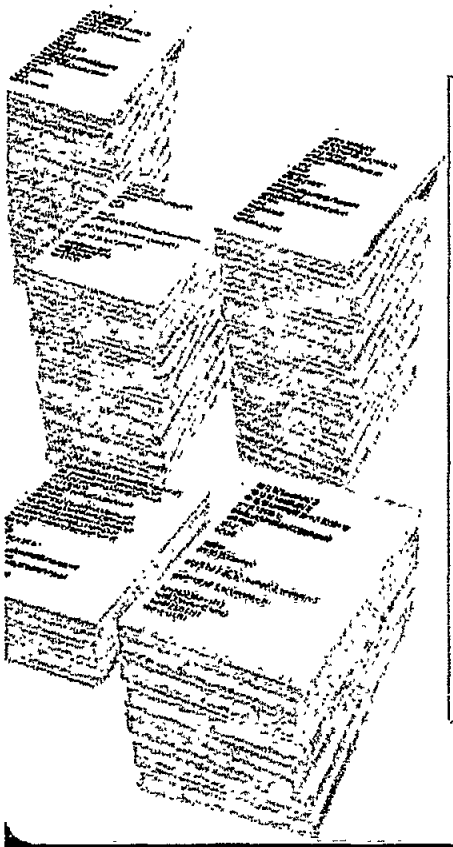


Notes

For example, there could be a function called WhatTime that provides the current time. For input, the WhatTime function takes a character, which is a single digit letter or number. That character represents the desired time zone. For output, WhatTime returns the current time in the desired time zone.

In this example, where the function is expecting to receive a time zone, it cannot be passed a dollar amount, such as $52.00, because a monetary input has no meaning to the WhatTime function.

Object Oriented Programming



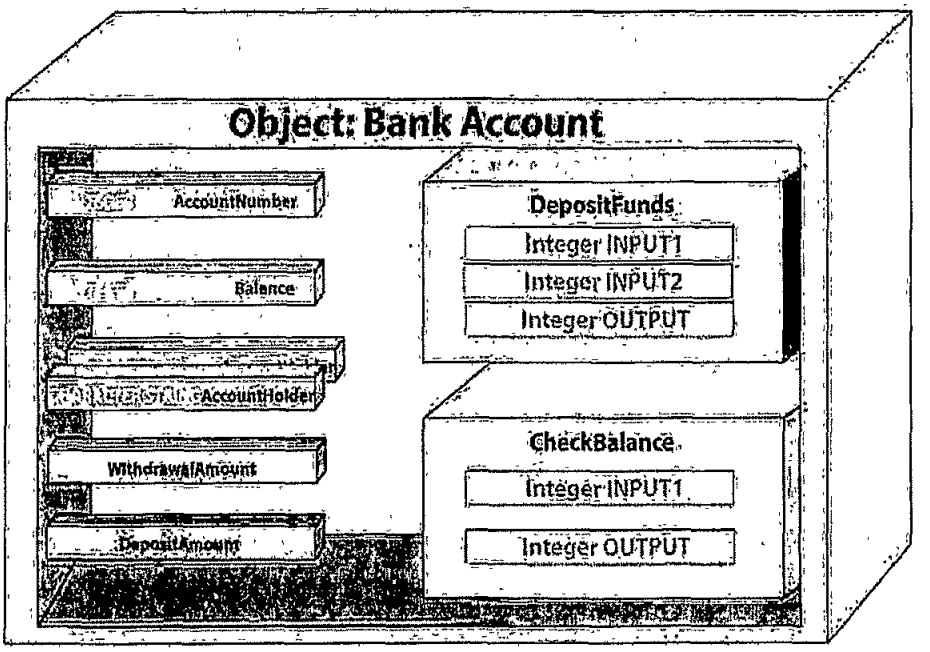
```
public class Account

      protected double balance;

      public Account( double amount )
balance = amount;

      public Account()
balance = 0.0;

      public void deposit( double amount )
balance += amount;

      public double withdraw( double amount )
if (balance >= amount)
balance -= amount;
                  return amount;
else
                  return 0.0;

      public double getbalance()
          return balance;
```


Notes

Another, more modern way to organize computer programs is known as object-oriented programming. Java is an example of an object-oriented programming environment.

One advantage of object-oriented programming is that it is a logical way for humans to think about and organize a program. Object-oriented programming groups functions and data together. Each object can be viewed as an independent little machine with a distinct role or responsibility.

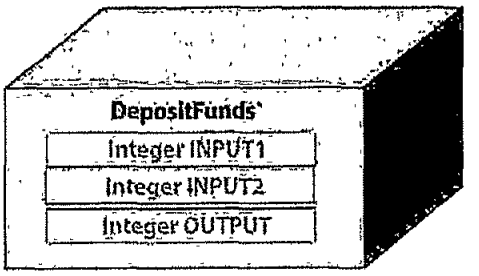
Object Oriented Programming



Notes

The pieces of data in an object are referred to as "fields." The functions or operations provided by an object are called "methods." For example, in a banking application, we would consider the bank account to be the object. Fields within that object might include: account number, account balance, names of the account holders, withdrawal amount and deposit amount. Each field has a data type. For example, the account number is an integer data type since it needs to hold a number – the account number. Similarly, the account balance is also an integer data type because it needs to hold a number representing the amount in the account. The Account Holder field, however, is a character string data type since it holds the name of the account holder, which is a collection of characters.

In this example, there are two methods provided by the BankAccount object: deposit funds and check balance. These methods are the operations that can manipulate the data in the BankAccount object.



(Integer)DepositFunds(Integer,Integer)

Method Signature

The MathWorks

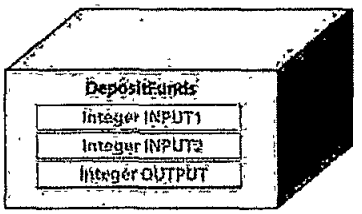
Notes

Each method has a method signature. The method signature includes the name of the method and defines what data types the method can accept as input.

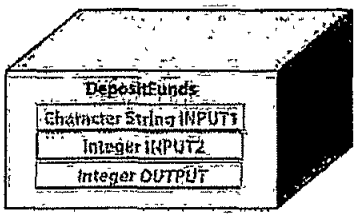
In object-oriented programming, an object can have two or more methods with the exact same name as long as there is some way to differentiate between those methods. There are two ways to differentiate between methods that have the same name.

Object Oriented Programming

Method Signature



(Integer)DepositFunds(Integer,Integer)

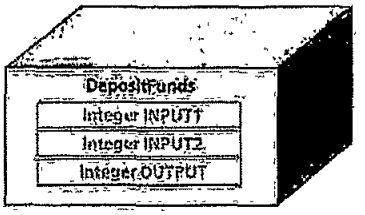


(Integer)DepositFunds(CharacterString,Integer)

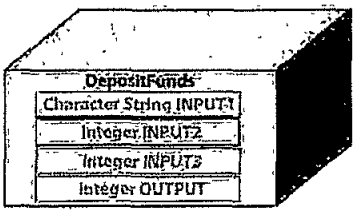
Notes

First, if the methods take different input data types, then there is a way to distinguish between the methods even though they have the same name. For example, there could be a second *DepositFunds* method in the BankAccount object. Assume that the second Deposit Funds method also takes two input parameters – one character string for the name of the account holder and one integer for the amount of the deposit. The two methods have the same name and the same number of input parameters, but one of the input parameters has a different data type. Because at least one input parameter has a different data type, it is possible to distinguish between the two methods even though they have the same name.

Method Signature



(Integer)DepositFunds(Integer,Integer)

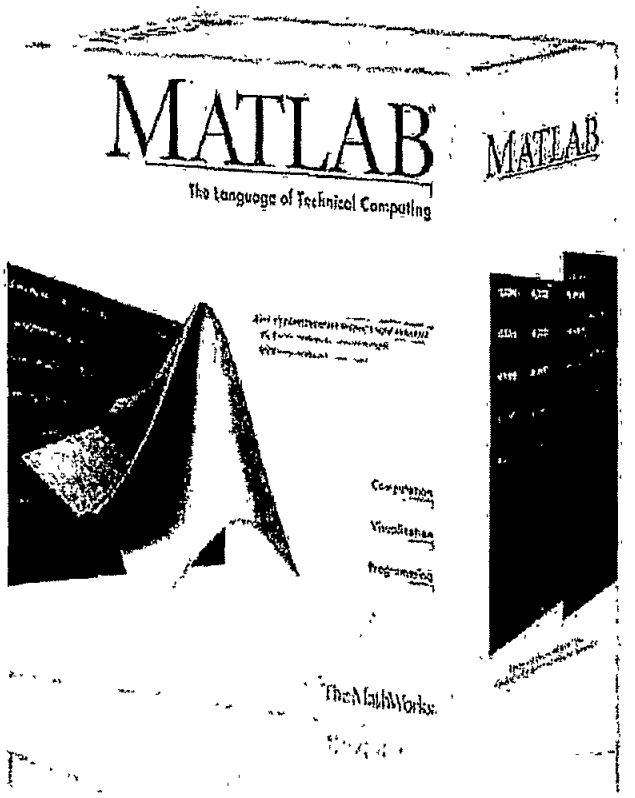


(Integer)DepositFunds(CharacterString,Integer,Integer)

Notes

The second way to differentiate between methods that have the same name is when each method takes a different number of input parameters. For example, there could be a second DepositFunds method that takes three input parameters – a character string for the name of the account holder, an integer for the account number and an integer for the deposit amount. Although the methods have the same name, it is possible to differentiate between them because they take different numbers of input parameters – one takes two input parameters while the other takes three input parameters. Therefore, these two methods can co-exist in the same object because it is possible to distinguish between them based on the number of input parameters.

What is MATLAB®?



The MathWorks

Notes

MATLAB is a computer program that provides a unique, interactive, high-level programming environment. Users can interact with MATLAB at a command line prompt, entering one command at a time. However, MATLAB users are not limited to entering commands at a prompt. Users can also create their own programs using MATLAB's programming environment. Most programs that customers create with MATLAB are organized functionally. However, MATLAB's data types are quite different from the data types provided by both typical functional programming environments and typical object-oriented programming environments.

What is MATLAB®?

Vector

[1 2 3 4]

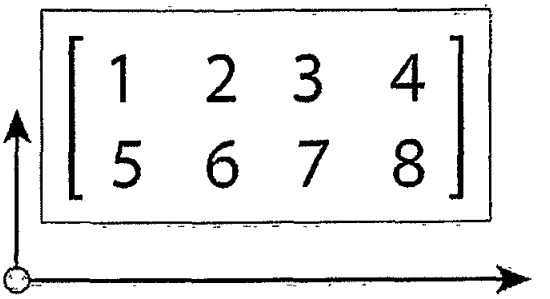
The MathWorks

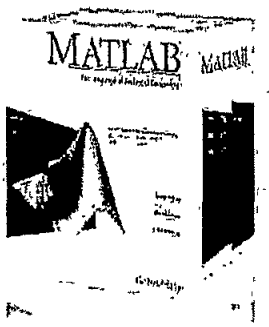
Notes

The basic data type used by MATLAB is an array or matrix. An array is a rectangular arrangement of data in one or more dimensions.

A one-dimensional array is commonly referred to as a vector. For example, this is a one-dimensional array of four elements. Because it is an array with one row and four columns, it is a 1 by 4 array.

What is MATLAB®?





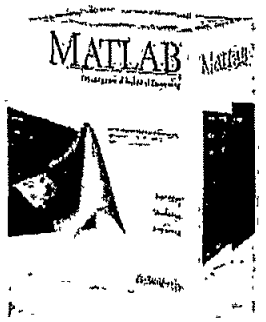
The MathWorks

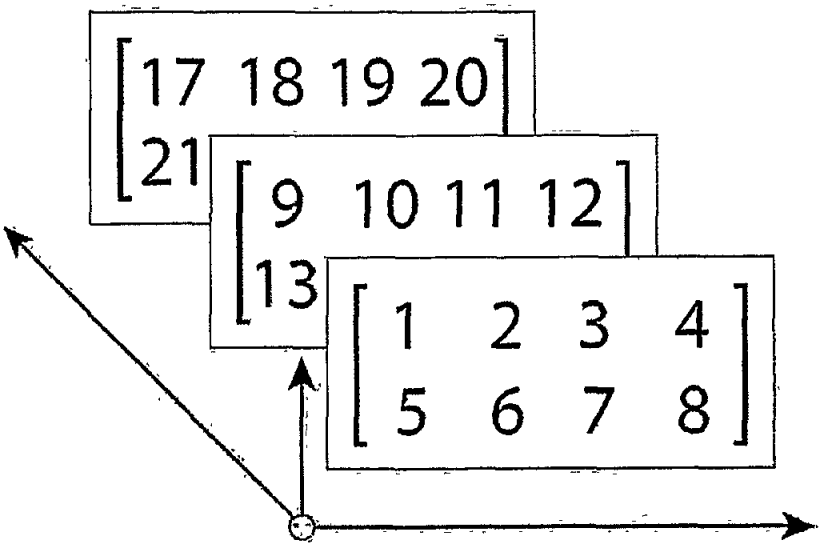
Notes

A two-dimensional array is commonly referred to as a matrix. For example, this is a two-dimensional array of eight elements. It is arranged in two rows and four columns. Therefore, it is a 2 by 4 array.

What is MATLAB®?

Multi-Dimensional Array





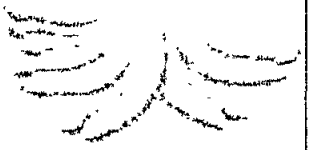
The MathWorks

Notes

Arrays can be any number of dimensions. For example, a three-dimensional array might have data for length, width and height. There can be four-dimensional arrays, five-dimensional arrays, six-dimensional arrays and so on.

MATLAB solves technical problems, including those that involve manipulation of arrays. Problems that are solved with linear algebra are one class of technical problems that are easy to solve using MATLAB.

How MATLAB® Is Used

			3 Apples 12 Bananas 1 Cantalope Total: $ 2.36
			12 Apples 2 Cantalopes Total: $ 5.26
			2 Bananas 3 Cantalopes Total: $ 2.77

Notes

Here is an example of how MATLAB might be used to solve a simple story problem.

Alice buys three apples, a dozen bananas, and one cantaloupe for $2.36.

Bob buys a dozen apples and two cantaloupes for $5.26.

Carol buys two bananas and three cantaloupes for $2.77.

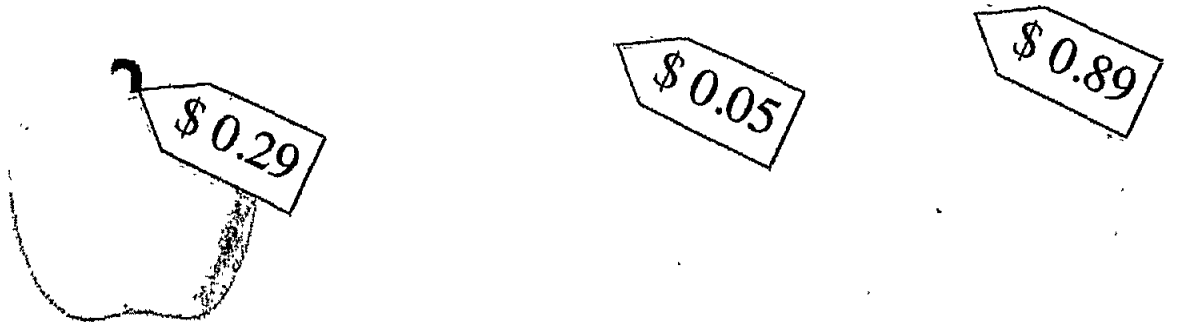
How much do single pieces of each fruit cost?

$$A=\begin{bmatrix} 3 & 12 & 1 \\ 12 & 0 & 2 \\ 0 & 2 & 3 \end{bmatrix} \qquad B=\begin{bmatrix} 2.36 \\ 5.26 \\ 2.77 \end{bmatrix}$$

Notes

This problem can be rewritten as a system of three linear equations with three unknowns, where A equals the cost of an apple, B equals the cost of a banana, and C equals the cost of a cantaloupe.

To solve this problem with MATLAB, a user can introduce a coefficient matrix to represent the left side of the three equations and a vector of prices to represent the right side of the three equations.



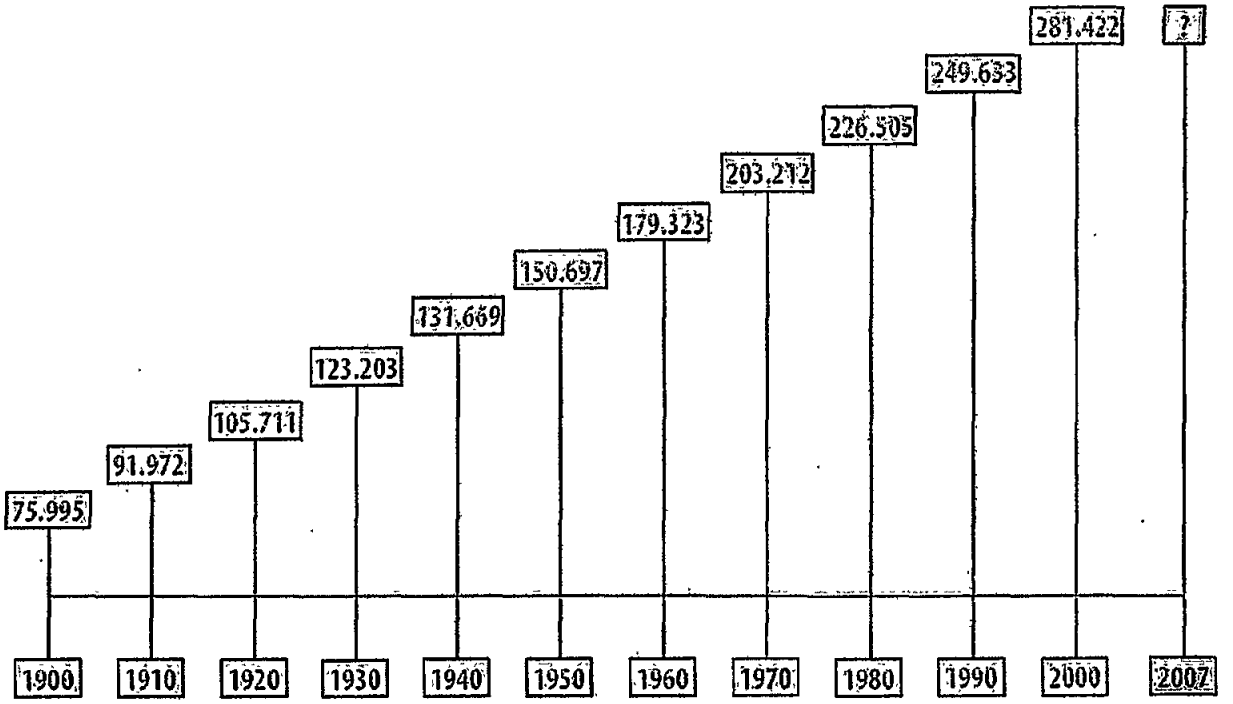
$$x = A \backslash B \qquad x = \begin{bmatrix} 0.29 \\ 0.05 \\ 0.89 \end{bmatrix}$$

Notes

The solution to the system of equations can easily be computed using the MATLAB "backslash" operator, which corresponds to division of the coefficient matrix by the vector of prices. This is a special kind of division that is appropriate for matrices and vectors and is not the division we learned in grade school.

This produces the following result, indicating that apples cost $0.29 each, bananas cost $0.05 each, and cantaloupes cost $0.89 each.

How MATLAB® Is Used, Part 2



Notes

Although that example was fairly simple, MATLAB can be, and is, used to solve more complicated, real-world problems. For example, MATLAB can be used to predict the current population of the United States based on past census data.

This is the United States census data for every ten years from 1900 through 2000. We wish to use this data to estimate the current population of the U.S.

How MATLAB® Is Used, Part 2

```
t = (0:10)'
p= [ 75.995  91.972  105.711  123.203  131.669  150.697  179.323  203.212  226.505  249.633  281.422 ]'
```

Equation Used To Model Over Time: $p(t) = c_1 t^2 + c_2 t + c_3$

The MathWorks

Notes

The Census Bureau stipulates that the data corresponds to April 1 of the census year. It is convenient to scale the time variable so that T equals zero corresponds to April 1, 1900 and one unit of time is one decade. The data can then be represented by two vectors.

A satisfactory mathematical model involves a quadratic polynomial, or parabola, in the time variable.

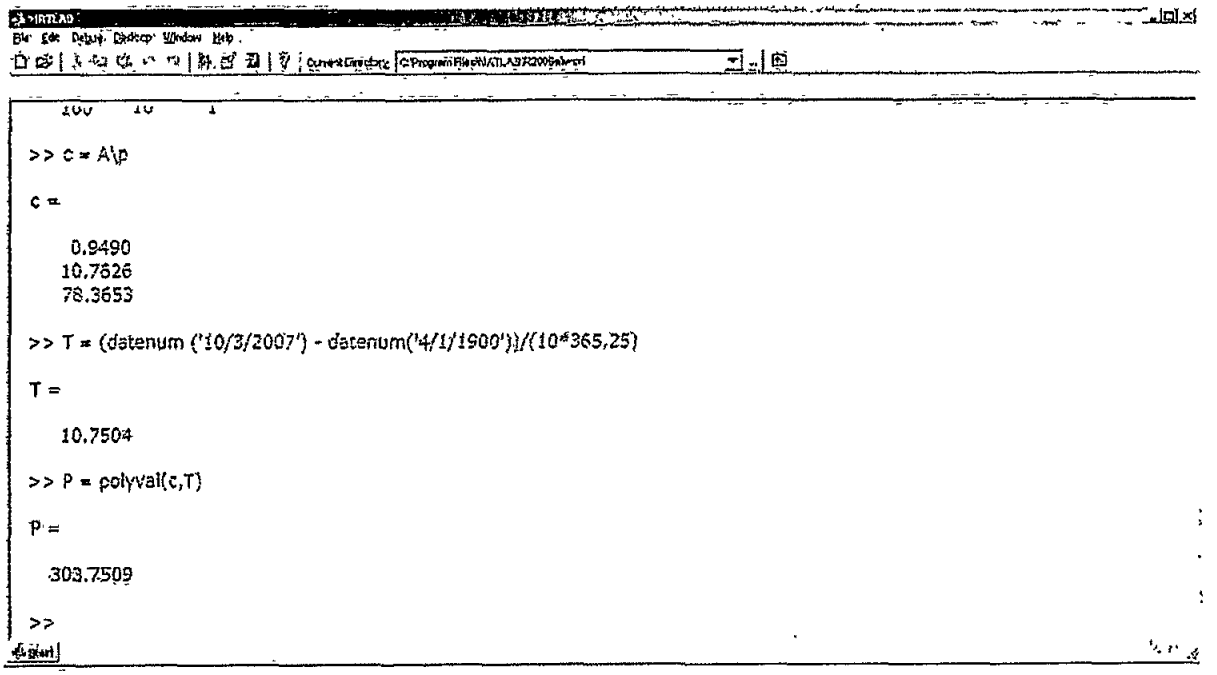
There are three unknown coefficients in this equation: C1, C2 and C3. The historical census data provides eleven data points that can be used to try to find values for the coefficients C1, C2 and C3. It is impossible to find coefficients so that the parabola curve matches the data exactly. Instead, we will do a "least squares" fit, where the coefficients are chosen to minimize the difference between the model prediction and the actual data points.

How MATLAB® Is Used, Part 2

t = (0:10)'

p= [75.995 91.972 105.711 123.203 131.669 150.697 179.323 203.212 226.505 249.633 281.422]'

Equation Used To Model Over Time: $p(t) = c_1 t^2 + c_2 t + c_3$



The MathWorks

Notes

This MATLAB statement creates a vector with eleven numbers – the numbers 0 through 10. This vector represents the value of t.

This MATLAB statement creates a vector that contains the eleven historical census data points.

This MATLAB statement creates an 11-by-3 matrix with elements that are powers of the time variable.

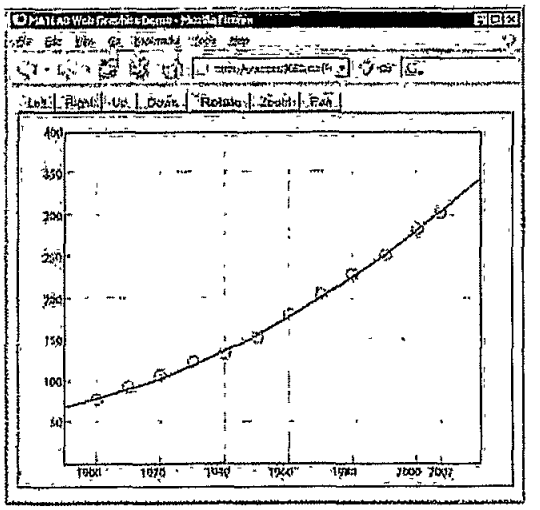
In this statement, the backslash operator computes a least squares approximate solution to eleven equations in three unknowns.

We choose October 3, 2007 as the date on which we will estimate the population in the US. We represent that date with this statement.

Finally, the polynomial with coefficients can be evaluated at time T with this statement.

The value returned by MATLAB for P is 303.7509. That represents an estimated population of approximately three hundred and three million people in the US on October 3, 2007.

How MATLAB® Is Used, Part 2

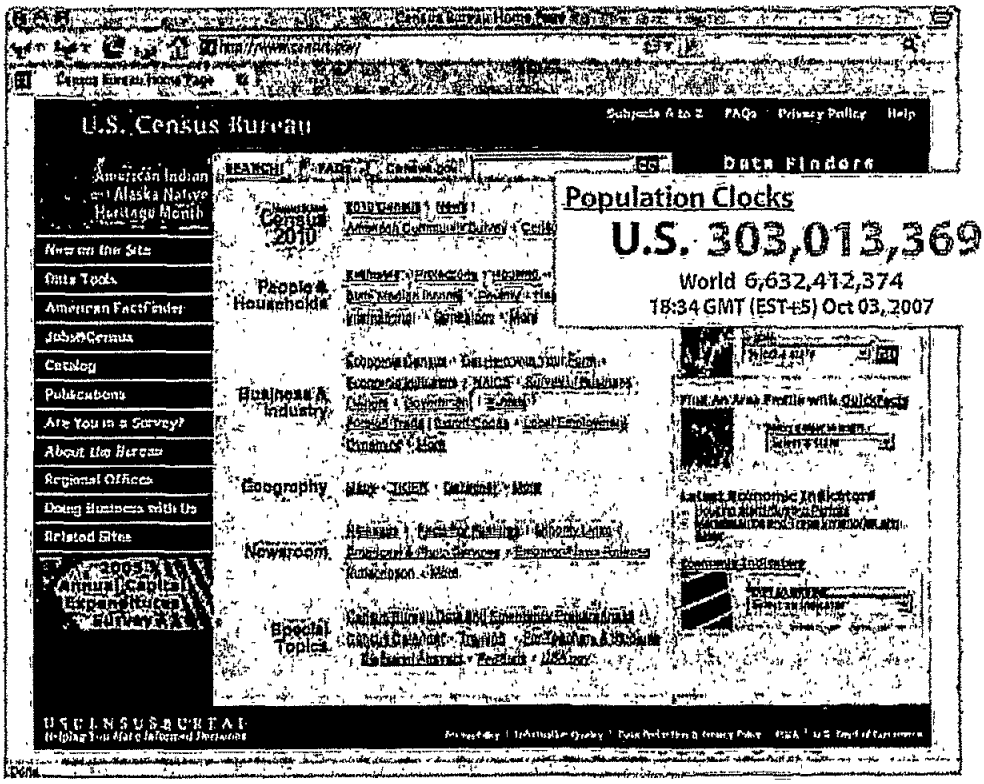


The MathWorks

Notes

In this graphic, generated using MATLAB, the first eleven red circles are the census data points, the solid blue curve is the parabolic fit, and the last green circle is the value MATLAB predicts for October 3, 2007.

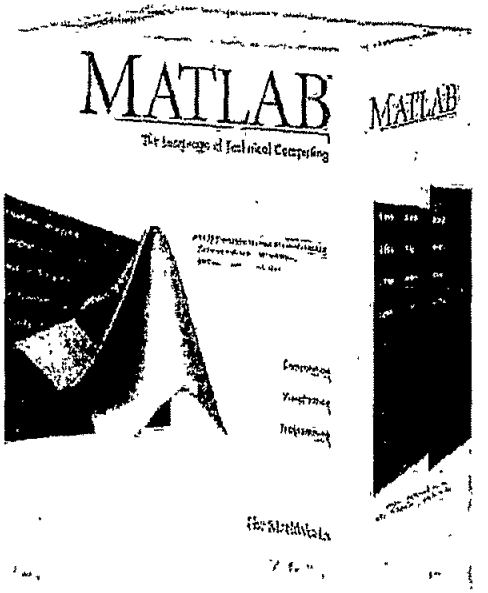
How MATLAB® Is Used, Part 2



Notes

The Census Bureau web site displays a population clock that uses more recent data and a more sophisticated mathematical model. However, the population clock returns a similar estimate of 303.0 million people.

How MATLAB® Is Used, Part 3



Notes

Although the Census example demonstrates the application of MATLAB to a potential real-world problem, even that example is relatively simple and involves only a few equations and unknowns. MATLAB customers routinely use MATLAB for much more complicated problems. For example, the Air Force and Navy use MATLAB to test their planes, including to evaluate engine thrust response and to calculate inlet recovery. Ford Motor Company uses MATLAB for sound quality analysis and calibration of the onboard diagnostics for its cars. Novartis uses MATLAB to model the uptake and distribution of a drug given to animals or humans. In these and other real-world problems, there can be hundreds and even thousands of equations and unknowns.

Problem Solved by the '338 Patent

Array-Based Mathematical Tool

Data Types

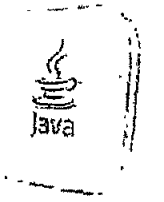
array of doubles

array of unsigned 8-bit integers

array of characters

Object-Oriented Environment

Data Types



The MathWorks

Notes

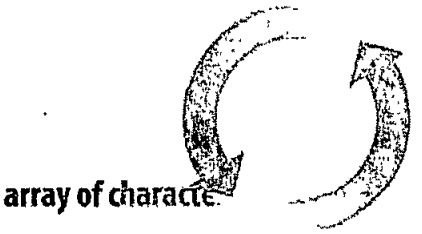
A user of an array-based mathematical tool such as MATLAB sometimes will wish to utilize a piece of code from an object-oriented environment, such as Java.

For example, Java offers a rich set of objects for creating graphical user interfaces. Allowing users of mathematical tools to access Java's objects would help users to create sophisticated graphical user interfaces for use with the mathematical tool. However, mathematical tool and object-oriented environments use different data types. Mathematical tools like MATLAB are array-based, meaning data types are represented as arrays, while object-oriented environments are not.

Problem Solved by the '338 Patent

Array-Based Mathematical Tool

Data Types

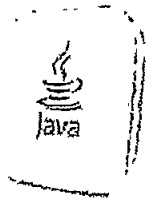


Object-Oriented Environment

Data Types

floats

shorts



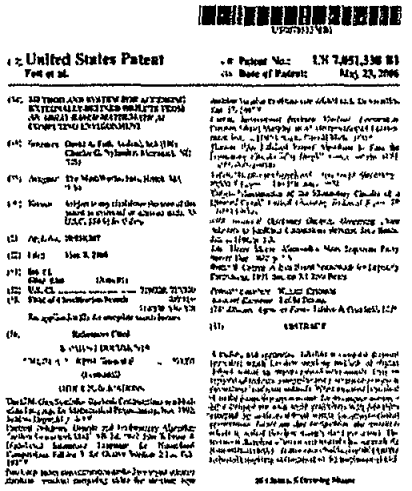
The MathWorks

Notes

Therefore, in order for a mathematical tool user to use a piece of code written in Java, the data that is passed to the Java environment must be translated into a data type that the Java environment recognizes and accepts.

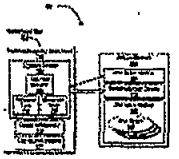
Similarly, once the Java code finishes executing, any data that is returned to the mathematical tool must also be translated back into an array that the mathematical tool recognizes and accepts.

Problem Solved by the '338 Patent



1. Translate mathematical tool data into data types that the object-oriented method will accept

2. Translate data returned by the object-oriented method into a data type that the mathematical tool recognizes

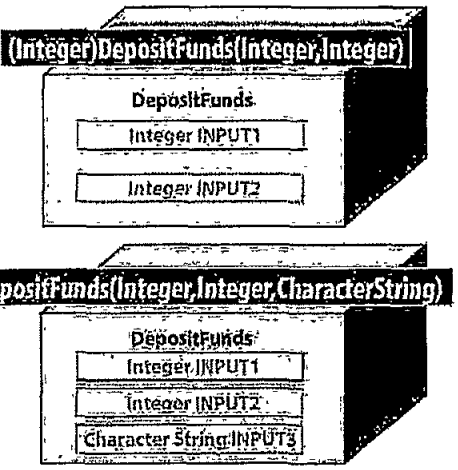


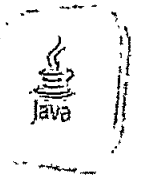
Notes

The '338 patent describes how to, first, translate mathematical tool data, which is stored as an array, into data types that the object-oriented method will accept; and second, to translate data returned by the object-oriented method into a data type that the mathematical tool recognizes.

Problem Solved by the '338 Patent, Part 2

```
a=1
b=1
c=1
int8(a)
int16(b)
int8(c)
a=35.25
b=89735893
c=DepositFunds (a, b)
```





Notes

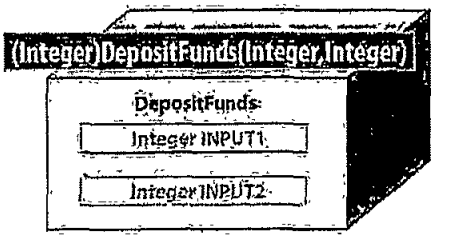
Additionally, certain object-oriented methods may not be capable of handling the data passed to it by a mathematical tool, even with translation. Therefore, the '338 patent also describes how to select an appropriate object-oriented method – one that is capable of handling data from the mathematical tool.

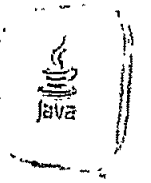
For example, assume there is a "Bank Account" object in Java. That object has two different methods that both have the same name. Both methods are named "DepositFunds".

The first DepositFunds method takes two input parameters, both integers. The second DepositFunds method takes three input parameters -- two integers and a character string. When the mathematical tool user wants to execute the DepositFunds Java method, the mathematical tool first has to determine which of the two DepositFunds methods is a better fit for the data the user wants to provide to the method.

Problem Solved by the '338 Patent, Part 2

```
a=1
b=1
c=1
int8(a)
int16(b)
int8(c)
a=35.25
b=89735893
c=DepositFunds (a, b)
```



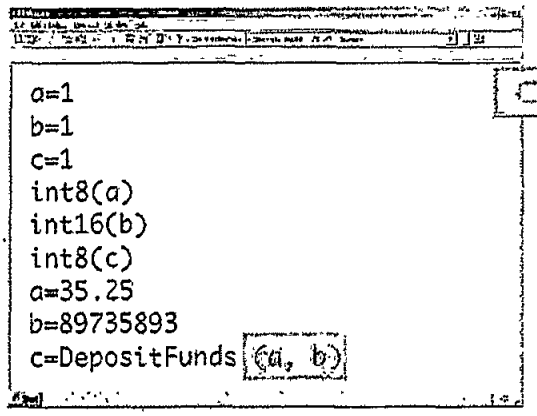


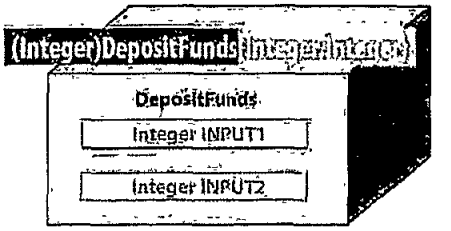
The MathWorks

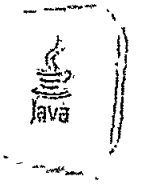
Notes

In this case, the user wants to provide two input parameters to the DepositFunds Java method -- an array with a single, signed 8-bit integer and an array with a single, signed 16-bit integer. The mathematical tool checks to see which DepositFunds methods are available in Java and finds the method signatures for the two methods. Only the first method has the right number of input parameters. Accordingly, the second method is rejected.

Problem Solved by the '338 Patent, Part 2







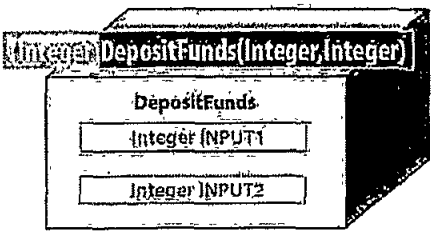
The MathWorks

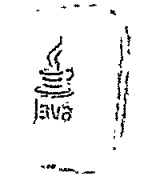
Notes

However, before the first method is determined to be the best fit, the mathematical tool must ensure that it can convert the mathematical tool data types into data types acceptable to the method as input parameters. In this case, the mathematical tool can convert the arrays into the integers required by the DepositFunds Java method, so that method is chosen.

Problem Solved by the '338 Patent, Part 2

```
a=1
b=1
c=1
int8(a)
int16(b)
int8(c)
a=35.25
b=89735893
c=DepositFunds (a, b)
```



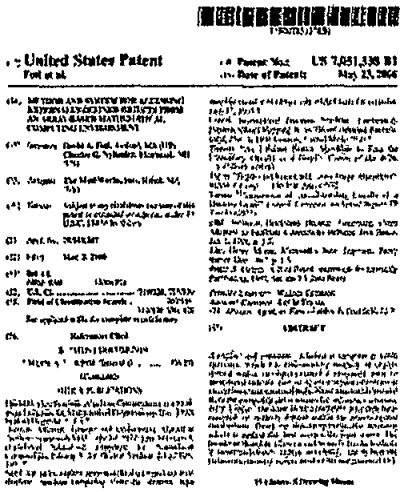


The MathWorks

Notes

After the DepositFunds method finishes executing in Java, it needs to return the Integer Balance to the mathematical tool. The mathematical tool translates that integer into a data type that the mathematical tool can accept, such as an array with a double precision floating point. Once the data is converted and returned to the mathematical tool, the mathematical tool has successfully called the method in Java.

Problem Solved by the '338 Patent, Part 2

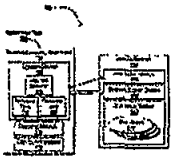


United States Patent

1. Translate mathematical tool data into data types that the object-oriented method will accept

2. Translate data returned by the object-oriented method into a data type that the mathematical tool recognizes

3. Allows two previously incompatible environments to interact and pass data in a seamless manner



The MathWorks

Notes

Thus, the '338 patent allows two previously incompatible environments – array-based mathematical tools and object-oriented environments – to interact and pass data in a seamless manner without the user having to translate the data types or choose the correct method.

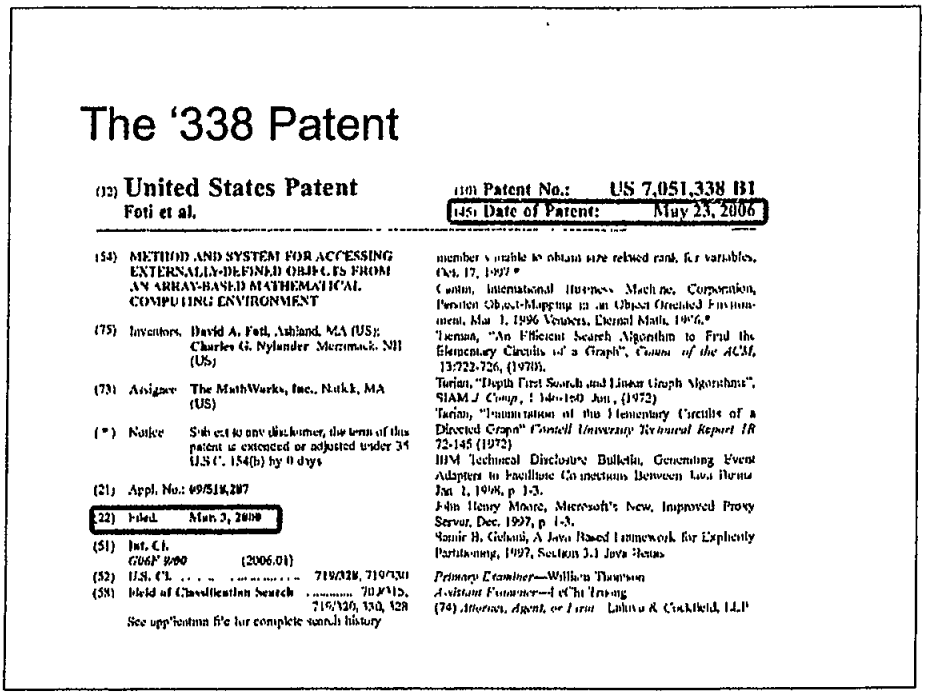
Defendant's Technology Tutorial

By COMSOL AB/COMSOL Inc.
Civil Action No. 6:06-cv-334-LED

This is a technology summary on US Patent 7,051,338 by COMSOL AB and COMSOL Inc.

Note that you can use the control buttons to stop, rewind or fast forward the presentation.

The '338 Patent



(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

member variable to obtain size related rank, for variables, Oct. 17, 1997.*
Cotton, International Business Machine Corporation, Persistent Object-Mapping in an Object Oriented Environment, Mar. 1, 1996 Ventura, Eternal Math, 1976.*
Tarjan, "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 Jan., (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

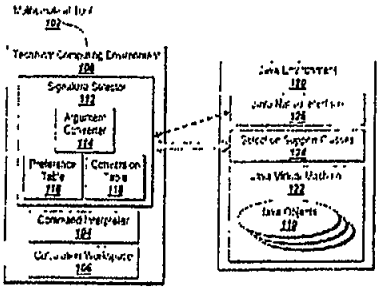
Primary Examiner—William Thomson
Assistant Examiner—Lechi Truong
(74) *Attorney, Agent, or Firm* Lahive & Cockfield, LLP

The patent application corresponding to US Patent 7,051,338 was filed on March 3, 2000. The '338 patent issued on May 23, 2006.

Overview of US Patent 7,051,338

(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.



According to the abstract, the patent covers "A method and apparatus, including a computer program apparatus, which facilitates 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."

Terminology, Summary

- Background Art: "Object-oriented programming"
  - Objects
  - Classes
  - Methods
  - Method signature
  - Method invocation
  - Overloading
- `338 Patent: "Array-based technical computing environment"

We will summarize some concepts pertaining to the method invocation.

We will start by describing some relevant background art concepts related to object-oriented programming including objects, classes, methods, method signatures, method invocation, and overloading.

Then we will describe the "array-based technical computing environment" disclosed in the '338 patent .

At the end of the tutorial we will explain how COMSOL Script version 1.1 and version 1.2 handle the problem solved by patent '338.

At various places in the tutorial we give examples to illustrate the technology and/or the definitions.

Background Art: Definitions of Terminology, Object-Oriented Programming and Objects

- "Object-oriented programming" is a programming paradigm that uses "objects" as building blocks to design computer programs. The '338 patent provides Java as an example of such an environment
- "Objects" are modules of computer code that specify the data types of a data structure and the kinds of operations (or "methods") that can be applied to the data structure. Each object is a unique instance, referred to as an instantiation of its corresponding "class". *See* 3:33-42 of the '338 patent

Object-oriented programming is a programming paradigm that uses "objects" as building blocks to design computer programs. The '338 patent provides Java as an example of such an environment. Object-oriented programming greatly reduces the resources needed to develop new applications.

The patent defines "Objects" as modules of computer code that specify the data types of a data structure and the kinds of operations, or "methods", that can be applied to the data structure. Each object is a unique instance, referred to as an instantiation of its corresponding "class".

Background Art: Definitions of Terminology, Classes and Methods

- "Classes" are prototypes that define characteristics of objects.
  See 3:36-39 of the '338 patent
- "Methods" are operations that can be applied to data.
  See 3:33-37 of the '338 patent

"Classes" are prototypes that define the characteristics of objects.

Using a metaphor, a class can be "dog" and the characteristics, common to all dogs, can be "coat" and "breed". In this metaphor, an object would be a specific dog, such as Lassie.

"Methods" are operations that can be applied to data.

Continuing with our metaphor, assume that "coat" has the value "long". Then applying the method "trim" would change the value of the characteristic "coat" to "short".

data can be the perception of a cat and the method can be "bark" and "run". methods can be "walk" and "bark" both functions that can be accomplished by a dog.

Background Art: Example of a Method

Method When(p)
  Input parameter "p" is of data type Time: HH:MM:SS

The method returns the time in a printable format

A method typically has several input parameters and a single output parameter of various types. The term "parameter" is used in the patent, while in object-oriented programming this term is usually referred to as "argument".

This example shows a definition of a method with the name "When" with a single input parameter, p.

The code executed at the invocation of "When" is left out in this example; only the definition of the name, number of input parameters, 1, and the type of the input parameter is demonstrated.

The type of the parameter, p, is of the data type Time, with hours, minutes, and seconds.

This method operates on input data to return the time in a printable format.

Background Art: Definitions of Terminology, Method Signature

- A "Method Signature" is a unique representation of the method's name, and the number and type of each input and output parameter of the method
  - Each method has a "Method Signature"
  - The method signature is used to distinguish between methods having the same name
  - *See* 3:45-49 of the '338 patent

Method signatures are used to represent available methods so that the methods can be examined and invoked.

According to the patent, a method signature is a unique representation of the method's name, the number of parameters, and the type of each input and output parameter of the method.

Background Art:
Example of a Method Signature

Method When(p)
Input parameter "p" is of data type Time: HH:MM:SS
The output parameter is of type string.

The method signature consists of its name, "When", its number of input parameters, 1, the type of the input parameter, Time: HH:MM:SS, and the type of its output parameter, of type string (the time in a printable format)

The method signature consists of the name of the corresponding method, "When", its number of input parameters, 1, the type of the input parameter, Time in hours, minutes, and seconds, and the type of its output parameter, the time in a printable format, a string.

Background Art: Definitions of Terminology, Method Invocation

- Methods in the object-oriented environment are requested and executed through a process called "invocation"
  - Methods are invoked by a request message specifying the name of the method, the input parameters, and the types of each input parameter
  - When more than one method has the same name the specific method to be invoked must be distinguishable by either:
    - the number of input parameters
    - the types of the input parameters

Methods in an object-oriented environment are requested and executed through a process called "invocation".

Methods are invoked by a request message specifying the name of the method, the input parameters, and the types of each input parameter.

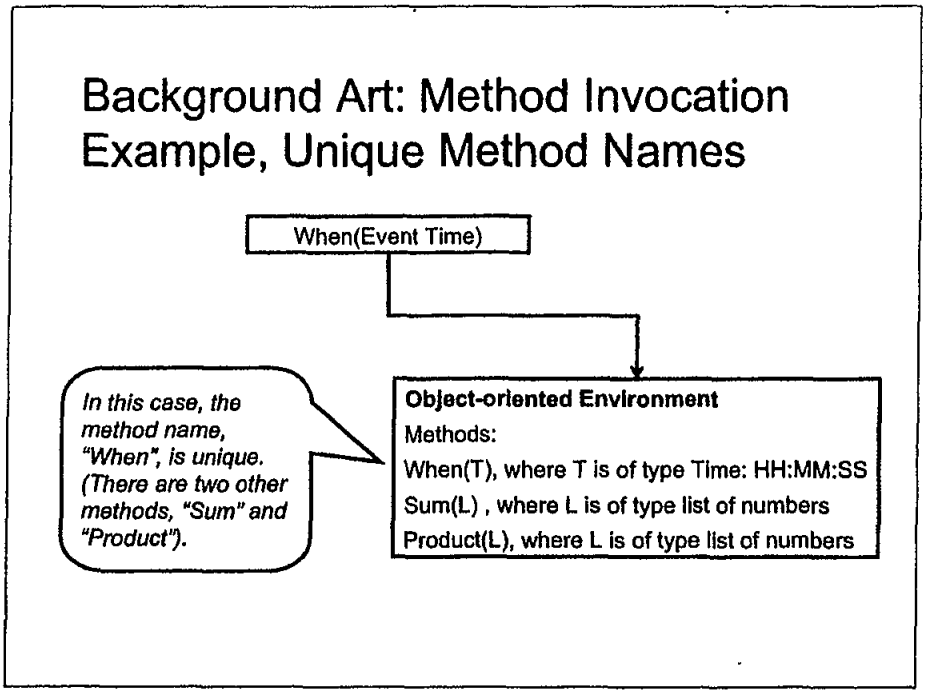
When two or more methods have the same name, the specific method to be invoked must be distinguishable by either the number of input parameters or the types of the input parameters. If such a distinction cannot be made, the appropriate method cannot be ascertained and an error will result.

Background Art: Definitions of Terminology, Method Invocation and Overloading

- Overloading describes the process of invoking a method when there are several methods with the same name but that differ in either the number of input parameters or the type of input parameters
- Overloading then distinguishes between the methods based on the parameters of the invocation
- Overloading was available in Java and C++ before the filing of the '338 patent

See: David Flanagan, "JAVA in a Nutshell", 3rd Edition, O'Reilly, 1999

Overloading describes the process of invoking a method when there are several methods with the same name, which differ in either the number of input parameters or the type of input parameters. Overloading then distinguishes between the methods based on the parameters of the invocation. Overloading was available in Java and C++ prior to the filing of the '338 patent.

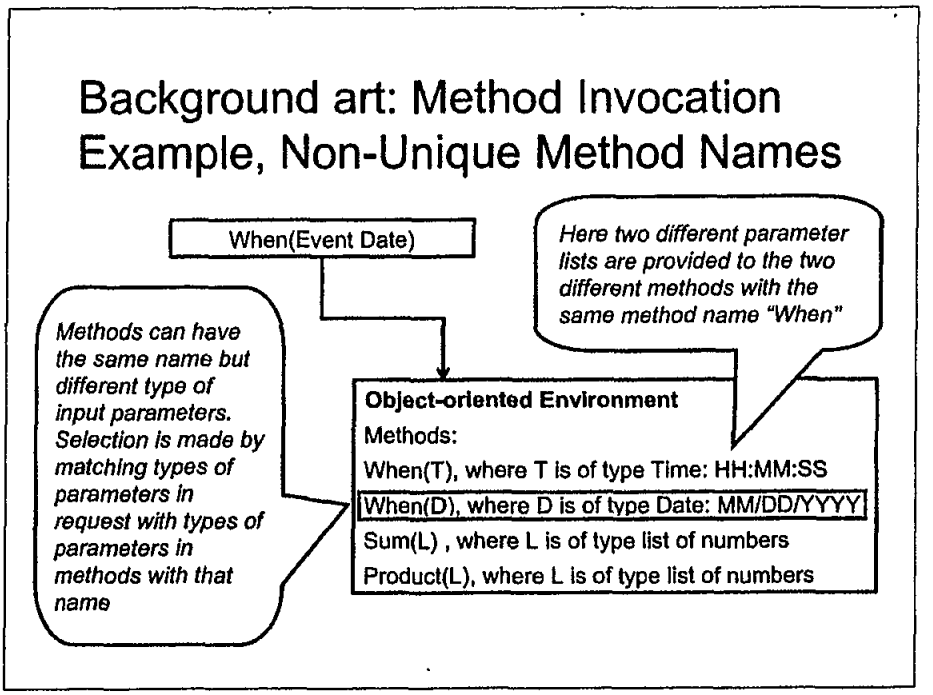


In this example there is an invocation request of a method named "When" in the upper rectangle.

In the object-oriented environment, the lower rectangle, there is only one method with that name defined.

If a method is requested and there is only a single method defined in the object-oriented environment with that method name, the task of determining what method to invoke is trivial.

In this example, there is only one method to choose from with the name "When". That one method will be invoked in response to the request.



In this example there are two methods defined with the same name, "When", both with one input parameter. Plus two other methods with other names, "Sum", and "Product".

The first signature corresponding to a method named "When" has the input parameter of type Time, and the second signature corresponding to a method named "When" has the input parameter of type Date.

Selection is made by matching types of input parameters in the invocation request with the input parameter data types in the method signatures.

As already mentioned, the well-known process of invoking one of several methods with the same method name, as in this example, is called "overloading."

In response to the invocation request in the upper rectangle, the method "When" with an input parameter Event of type Date is invoked, which matches the second signature.

So the second "When" in the definition list is invoked and executed.

Background art: Method Invocation Example, The Application

```
If (Event Date = Current Date)
  Print (When (Event Time))
Else
  Print (When (Event Date))
```

Here's an example of how the two methods with the same name can be invoked from an application software.

This application code prints the Event Time if it is executed the same day as the event occurred and the Event date if it is executed a different date than the event occurred.

For example, when displaying the time of arrival of an e-mail message, it is convenient to display the date for older messages while the time is displayed for messages that arrived today. This application will call one of the two methods named "When", described above, depending on the type of the input parameter in the method invocation request.

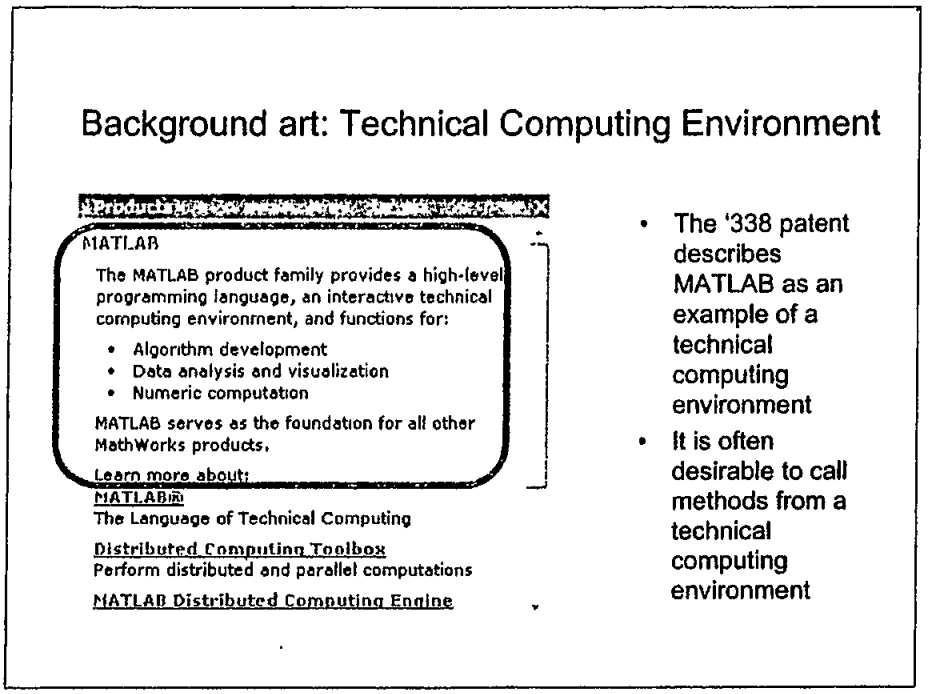
Current Date returns the date at execution time.

Recall from the example above:

the input argument "Event Time" is of data type Time: HH:MM:SS

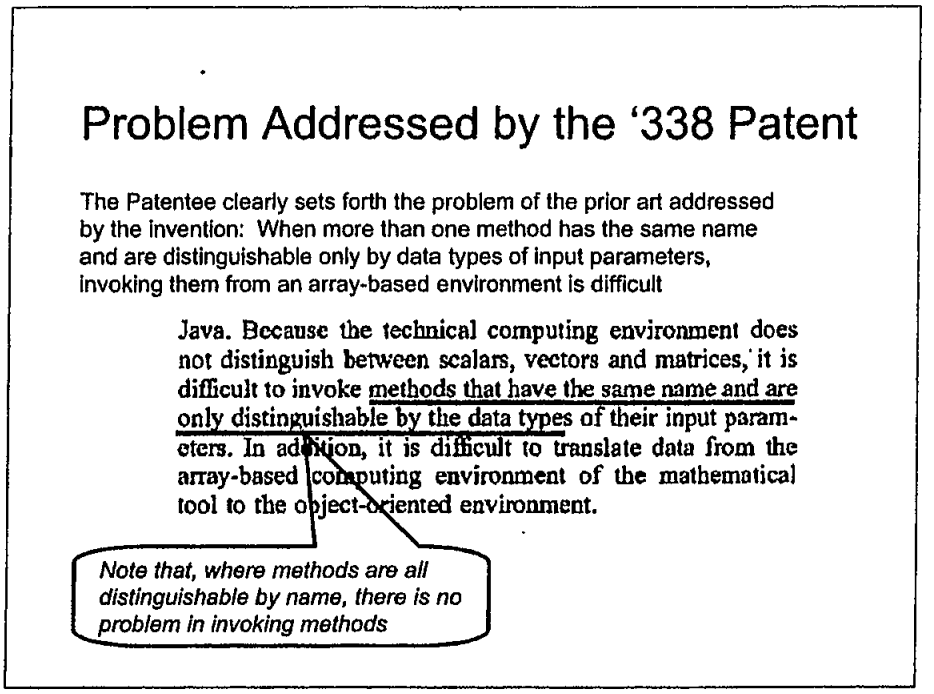
the input argument "Event Date" is of data type Date: MM/DD/YYYY

Therefore, when executing this snippet of code, the object-oriented environment will select the appropriate method named "When" from the two methods named "When" based on the type of input parameter in the method invocation request resulting from the code snippet.



The '338 patent describes the MATLAB product as an example of a technical computing environment. Such environments are used by engineers and others for various functions, such as to analyze data and model complex physical systems.

It is often desirable to call methods in an object-oriented environment from a technical computing environment. For example, a user of MATLAB may want to take advantage of user interface capabilities of the Java language by calling methods in Java.



The Patentee clearly sets forth the problem of the prior art addressed by the invention of the '338 patent: When more than one method has the same name and are distinguishable only by data types of input parameters, invoking them from an array-based environment is difficult. In other words, the known mechanism of overloading is not easily utilized when the method invocation request originates in an "array-based" technical computing environment.

Note that, where methods are all distinguishable by name, there is no problem in invoking methods.

Definitions of Terminology, Array-Based Technical Computing Environment

- Mathematical tools are often "array-based"
  - Data types are represented as two-dimensional arrays
  - Do not distinguish between scalar, vector, or matrix
- It is difficult to interface the technical computing environment to an object-oriented environment, such as Java
  - It is difficult to invoke methods that have the same name and are only distinguishable by the data types of their input parameters
- *See* 1:42-55 of the '338 patent

The '338 patent defines an "Array-Based Technical Computing Environment" by saying that Mathematical tools are often "array-based" and that Data types are represented as two-dimensional arrays.

These two-dimensional arrays do not distinguish between a scalar, vector, or matrix.

The invention disclosed in the `338 patent addresses the problem of difficulty in invoking methods in an object-oriented environment from an array-based environment when multiple methods have the same name and are only distinguishable by the data types of their input parameters.

Scalars, Vectors, and Matrices

- scalar: the number 7
- vector: the list, or one-dimensional array, of numbers

  1, 25, 4

- matrix: the two-dimensional array, or table, of numbers

  3, 6, 17
  7, 2, 21

Let's explain the mathematical concepts of a scalar, a vector, and a matrix.

A scalar is just a number, like 7

A vector is a list, or a one-dimensional array, of numbers, like 1, 25, 4

And a matrix is a table, or a two-dimensional array, of numbers, like, 3, 6, 17, and on a new row, 7, 2, 21

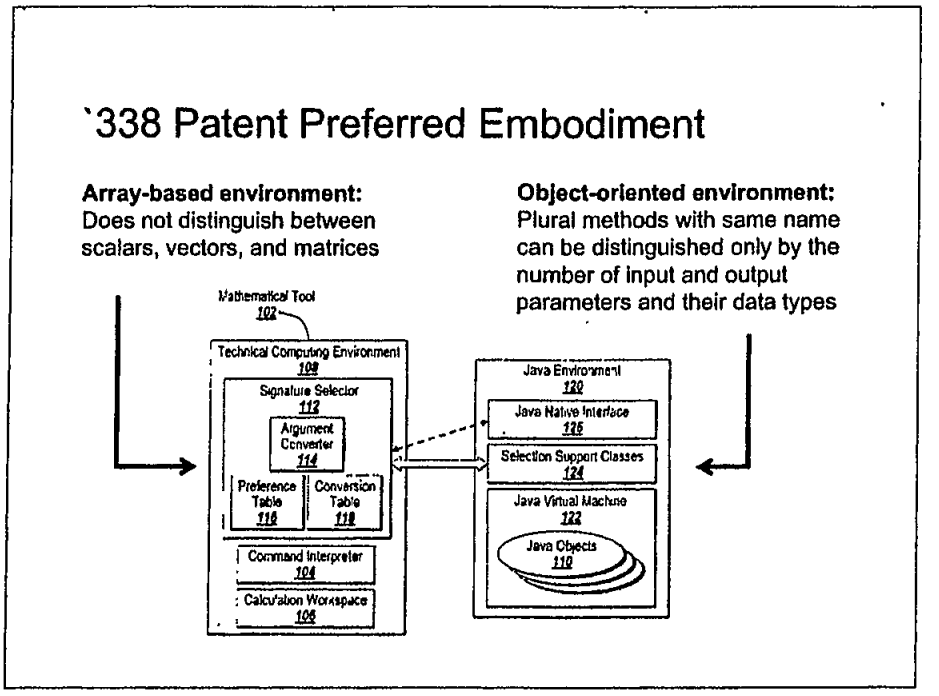
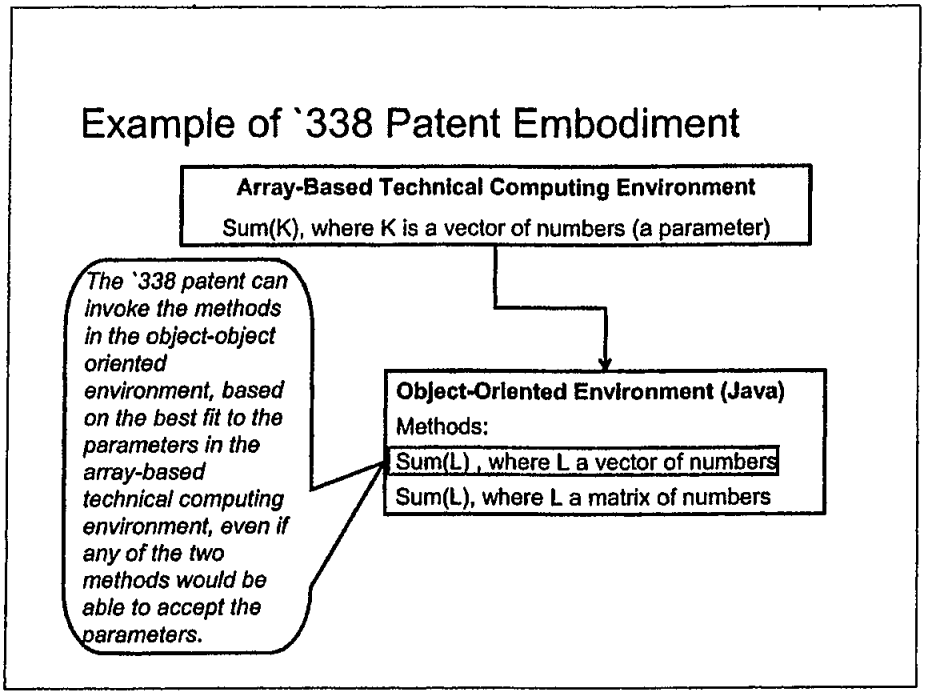


Figure 1 in the patent shows the array-based environment to the left and the object-oriented environment, Java, to the right.

The array-based environment does not distinguish between scalars, vectors, and matrices.

The object-oriented environment can have plural methods with the same name, which can be distinguished only by the number of input and output parameters and their data types.

The patent thus describes an overloading mechanism suitable for an array-based environment with scalar, vector, and matrix data types, for invoking methods in an object-oriented environment, such as Java.

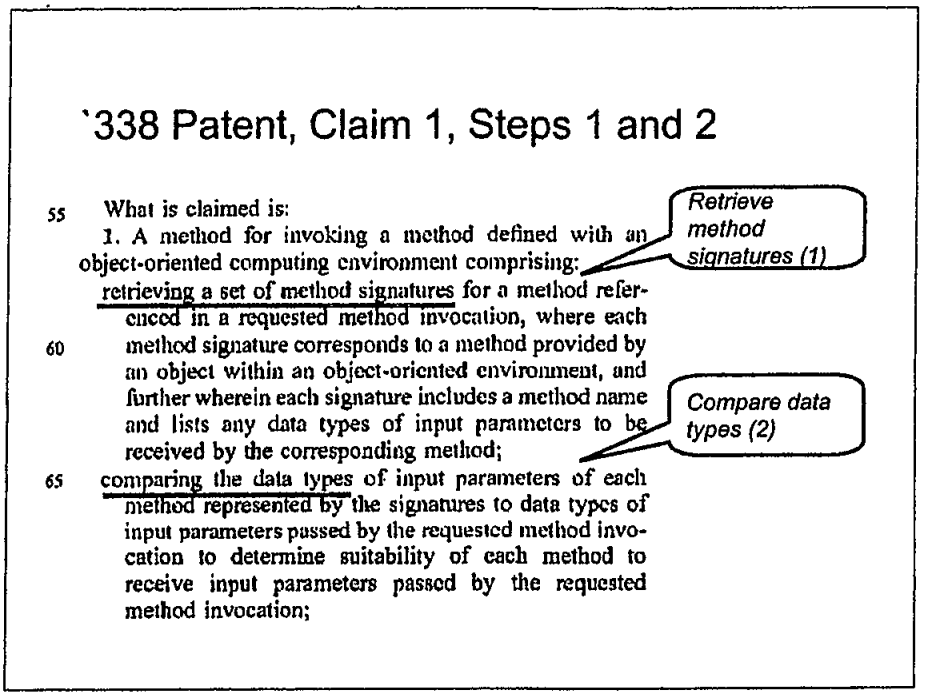


This is an example of how the method and system disclosed in the `338 patent works.

The array-based technical computing environment requests invocation of a method named "Sum" through a method request having the method name and an input parameter "K" that is a vector, which can also be viewed as a one-dimensional array of numbers.

There are two methods in the object-oriented environment that would apply, that is, two methods named "Sum". Both these methods would be able to compute a "sum" of the numbers in the vector.

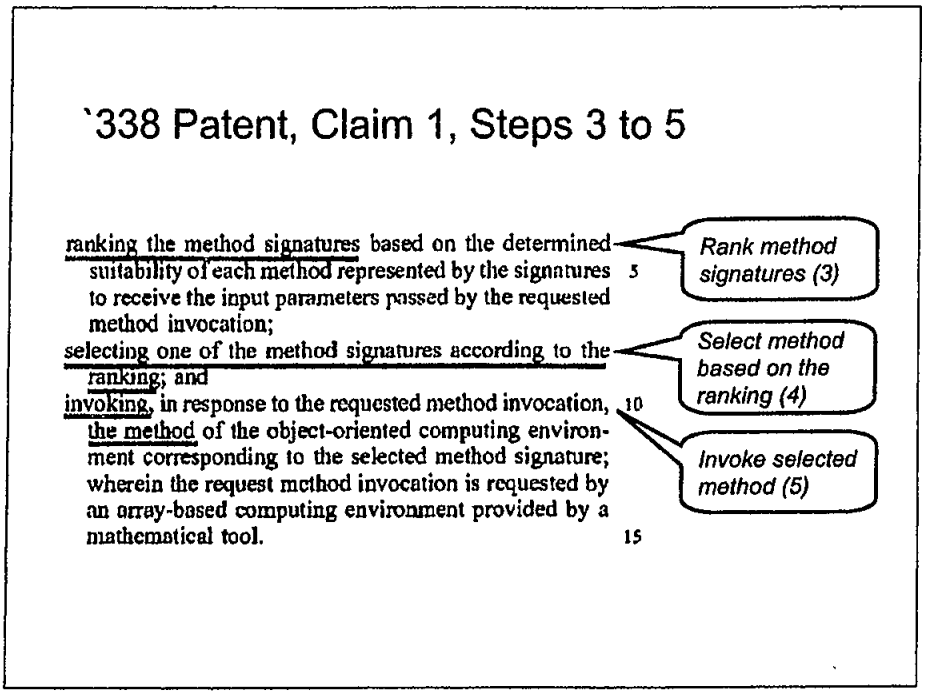
The system disclosed The `338 patent can invoke the methods in the object-oriented environment, based on the best fit to the input parameters passed by the method invocation request from the array-based technical computing environment.



The steps recited in Claim 1 in the '338 patent can be summarized as follows:

As a first step, a set of plural method signatures are retrieved for a method referenced in a requested method invocation. In other words, all method signatures with a name that is identical, or similar, to the name specified in the request are retrieved as noted at column 4, lines 57-60.

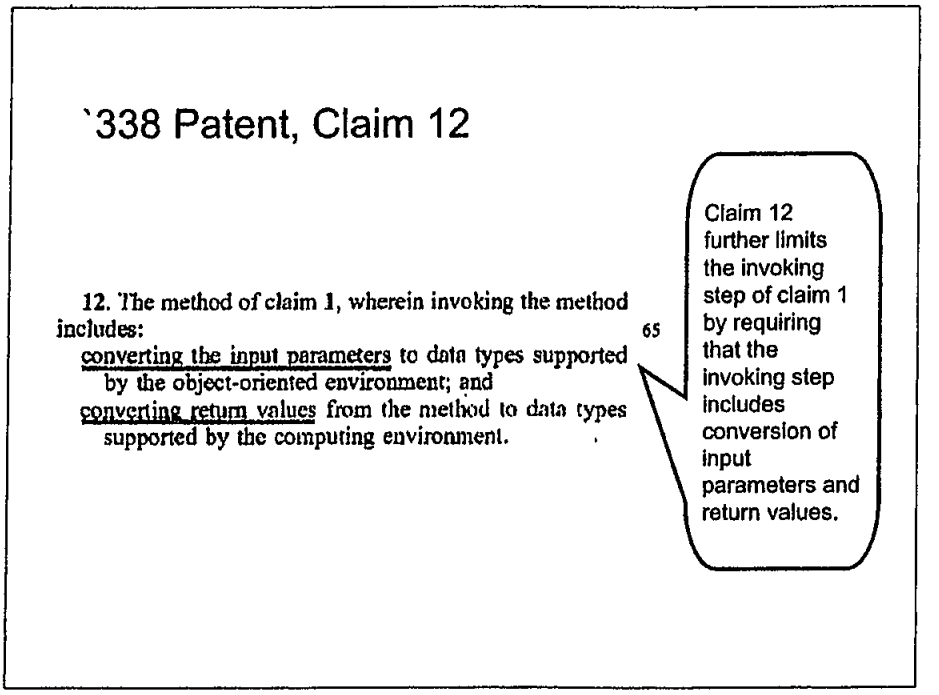
As a second step, compare the data types of input parameters of each method, represented by the signatures, to data types of input parameters passed by the requested method invocation. This comparison is for the purpose of determining the suitability of each method to receive input parameters passed by the requested method invocation.



As a third step, the retrieved method signatures are ranked based on the determined suitability.

As a fourth step, one of the plural method signatures is selected according to the ranking.

Finally, as a fifth step, in response to the requested method invocation, the method in the object-oriented computing environment corresponding to the selected method signature is invoked.



Claim 12 further limits the invoking step of claim 1 by requiring that the invoking step includes conversion of input parameters and return values. Claim 12 implies that the invoking step includes executing the method because return values are the result of method execution.

COMSOL Script

All COMSOL Multiphysics modeling capabilities are available through COMSOL Script. It lets you interact with models through an interactive graphical user interface, the COMSOL Desktop, for just about any analysis purpose you can think of.

COMSOL Script includes more than 600 high-level commands for data analysis and visualization. Simply save a COMSOL Multiphysics model as a Model M-file and then extend your simulations and analysis. Tailor-make your modeling through building customized user interfaces for others to access your modeling work.

COMSOL Script is a mathematical tool that can leverage Java methods

COMSOL Script V1.1 and V1.2 Method Invocation

- COMSOL Script provides a method declaration for each method that can be called from the technical computing environment
- There may be only one method declaration per method name and number of input parameters
- When a method is to be invoked only the method name and number of parameters in a method request from the technical computing environment is used to select a method
- After a method has been selected the method is invoked. As part of invocation the method input parameters are converted from their data types in the technical computing environment to the data types of the object-oriented environment

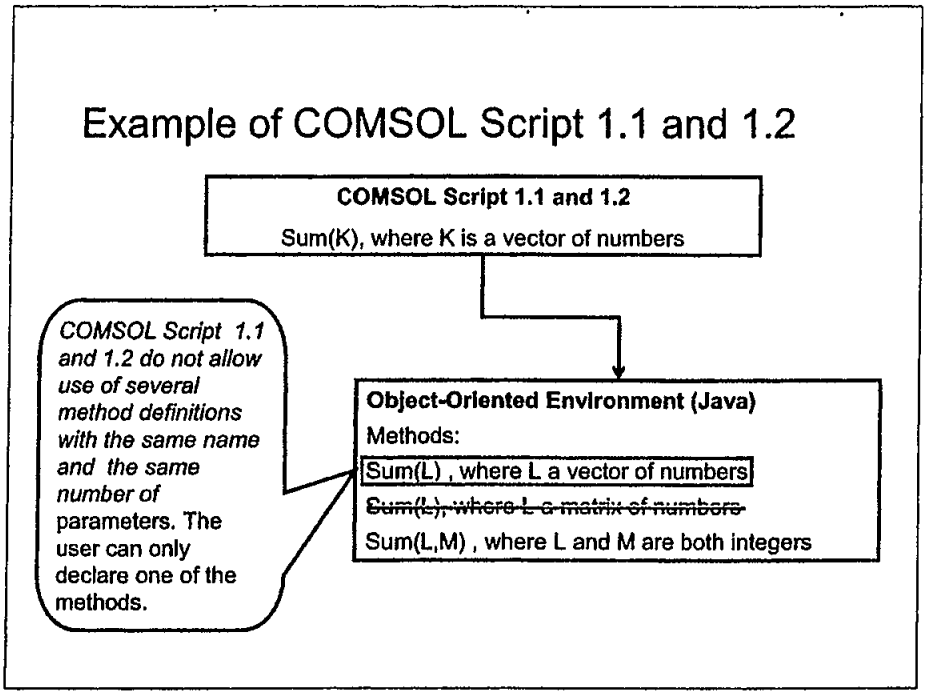
This is how COMSOL Script version 1.1 and version 1.2 determine which method to invoke.

COMSOL Script provides a method declaration for each method that can be called from the technical computing environment.

There may be only one method declaration per method name and number of input parameters.

When a method is to be invoked, only the method name and number of parameters in a method request from the technical computing environment is used to select a method.

After a method has been selected, the method is invoked and the method input parameters are converted from their data types in the technical computing environment to the data types of the object-oriented environment.



Here is an example of method invocation from COMSOL Script 1.1 and 1.2 to Java methods.

COMSOL Script 1.1 and 1.2 do not allow use of several method definitions with the same name and the same number of parameters or arguments. In COMSOL script only two of the three methods, all named "Sum", in this example can be declared and thus invoked from COMSOL Script 1.1 and 1.2. Any overloading is resolved by reference to the number of input parameters. Note that the first and third methods shown above, the two valid methods, have different numbers of input parameters. The first method has one input parameter and the second method has two input parameters.

In this example, since the request has one input parameter, K, the method with one input parameter is selected for invocation in response to the request. Note that there is only, and can only be, one method that can correspond to the request.

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

THE MATHWORKS, INC.	)	
	)	DOCKET NO. 6:06cv334
	)	
-vs-	)	
	)	Tyler, Texas
	)	9:30 a.m.
COMSOL, ET AL	)	February 6, 2008

TRANSCRIPT OF MARKMAN HEARING
BEFORE THE HONORABLE LEONARD DAVIS,
UNITED STATES DISTRICT JUDGE

A P P E A R A N C E S

FOR THE PLAINTIFF:

MR. TOM JACKSON
MR. TERENCE MURPHY
JONES DAY
2727 N. Harwood
Dallas, Texas 75201-1515

MS. KRISTA SCHWARTZ
JONES DAY
North West
Wacker Dr., Ste. 3500
Chicago, IL 60601

MS. SUSAN GERBER
JONES DAY
North Point
901 Lakeside Ave.
Cleveland, OH 44114

COURT REPORTER:

MS. SHEA SLOAN
211 West Ferguson
Tyler, Texas 75702
903/590-1176

Proceedings taken by Machine Stenotype; transcript was produced by a Computer.

THE COURT: Jump back over to "fitness ranking." Where does COMSOL come up with the numerical value definition?

MR. WATKINS: Judge, I don't really think -- the more I have thought about that I don't much care. I think we could put numbers, we could put letters. Putting the numerical one in there is probably on overreach by us and so I am kind of okay with theirs because the way I look at this, Judge, is you could have a list of fitness signatures -- of method signatures and you could rank them 1, 2, 3, 4. You could rank them A, B, C, D: and that would work because our minds know how that works. But if you assign them to a class and a class with red, yellow, and blue and you assigned each of these method signatures to a class, how are you going to select which one is the best? Because there is no ranking order between red and blue and yellow. It doesn't tell you anything, so you have to find out some way of ranking them.

I think they are right when they say it doesn't necessarily have to be numbers, it could be letters, it could be any kind of system in which the computer could recognize the relative value of the method signatures to each other based on their already determined suitability.

THE COURT: All right. If you are backing away from your proposed construction, are you saying it doesn't need construction, or are you endorsing their -- agreeing to their proposed --

MR. WATKINS: Once we determine what "ranking" means, I don't think "fitness ranking" needs any construction.

THE COURT: All right. Response?

MS. SCHWARTZ: We are happy to accept their position that "fitness ranking" doesn't need construction. I still think it is important to understand what "fitness ranking" is because what Mr. Watkins is arguing is that you have to have this ordering by levels in order to be able to do a selection. First of all, there is no restriction on how a method is selected in the specification or the claims. It says you select one. It doesn't say you have to select the highest. It doesn't say you have to select one by any given means. They are trying to restrict the scope of the claims beyond what is in the claims and the specification, so how a method is selected is irrelevant.

If I assign methods to -- let's say I find three methods that are suitable and I choose one of those three at random, I am still selecting one and I still have ranked those. There is no requirement anywhere in the specification, the file history, the claims that requires that you have to choose the highest ranking one.

THE COURT: Well, what does that do to your "ranking" step, though? Doesn't it render it meaningless if you can select one randomly?

MS. SCHWARTZ: No, it doesn't. You have to make

sure the method will work. I have a method in an object-oriented environment that let's say it takes an eight-bit character value. And in my mathematical tool I am trying to pass to at a floating point this big large number. There is no way that I can take that big large number and transform it to fit into an eight-bit character. It just won't work. So I have to be able to rank that. I have to compare and see this isn't going to work. I rank it as not suitable, and I tell the user this one is not going to work.

But there is an important purpose in being able to evaluate each method and see if it is even able to accept the data that the user wants to pass to it. Whether you go ahead and rank it down more specifically in terms of this one is best able to receive that data or whether you just have a category of these are all able to receive the data, the claims don't require that level of specificity.

If we look at the doctrine of claim differentiation, that has to be because Claim 2 where we have "fitness ranking" is where you start putting things in an order where you have different levels of suitability. With the doctrine of claim differentiation, Claim 1 and ranking has to be broader because the "fitness ranking" term is directly dependent on ranking. So according to Federal Circuit precedent, that has to have a narrower scope. So if the levels of suitability are what "fitness ranking" is, then "ranking" has to be something

bigger, so it has to encompass just being able to look at methods and say, well, will this one work or won't. So having two classes of suitable or not suitable as opposed to how suitable the method is.

I think that is the important point that COMSOL doesn't address in today's hearing or their brief is their construction gives the same meaning to "fitness ranking" and "ranking."

THE COURT: What is your response to that on the claim differentiation?

MR. WATKINS: I think the Court has already indicated the problem. They have either got to decide that they violated the rule of claim differentiation or they have not enabled their patent because they haven't told us how they are going to select from this list if they don't rank in the way that we mean ranking. I do think they have a serious problem with claim differentiation. Claim differentiation is simply one of a number of rules of construction that we use. But that doesn't mean we have to alter the word "ranking" from what its ordinary common term as it means and from the way that this patentee has used it in its claims, in its specification, and in front of the Patent Office to change that in order to save them from their problem they have got with claim differentiation.

Now, if the Court adopts theirs, everything violates

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.	)	
	)	

NOTICE OF APPEAL

Notice is hereby given that The MathWorks, Inc. ("MathWorks"), plaintiff in the above named case, hereby appeals to the United States Court of Appeals for the Federal Circuit from the final judgment entered in this action on the 10th day of March, 2008.

Dated: March 24, 2008

Respectfully submitted,

/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.

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 24th day of March, 2008.

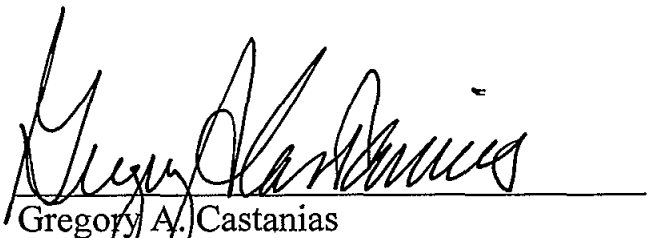
/s/ Krista S. Schwartz

PROOF OF SERVICE

I hereby certify that on August 15, 2008, two bound copies of the foregoing APPENDIX were served by overnight mail through a third-party commercial carrier (Federal Express) upon the following principal counsel:

Thomas H. Watkins, Esq.
Brown McCarroll, LLP
111 Congress, Suite 1400
Austin, TX 78701

I also certify that on August 15, 2008, twelve bound copies, including the original, of the foregoing APPENDIX were filed, by hand delivery, in the Office of the Clerk, United States Court of Appeals for the Federal Circuit.



Gregory A. Castanias
Attorney for Plaintiff-Appellant