

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

**Valtrus Innovations Ltd. and
Key Patent Innovations Ltd.,**

Plaintiffs

v.

SAP America, Inc. and SAP, SE.

Defendants.

§
§
§
§
§
§
§
§
§
§
§

CIVIL ACTION NO. 2:24-CV-00021-JRG

JURY TRIAL DEMANDED

DECLARATION OF MARTIN FARACH-COLTON, M.D., PH.D.
IN SUPPORT OF PLAINTIFFS' PROPOSED CLAIM CONSTRUCTIONS

Table of Contents

I.	Introduction and qualifications.....	1
A.	Education	1
B.	Career History	1
C.	Publications and Patents	2
D.	Other Relevant Qualifications.....	3
E.	Materials and Other Information Considered	4
F.	Compensation	4
G.	Prior Testimony	4
II.	Summary of opinions	4
III.	Understanding of the Law.....	5
A.	Claim Construction	5
B.	Means-Plus-Function Claims.....	7
C.	Indefiniteness	8
IV.	The '244 Patent	9
A.	Overview.....	9
B.	Level of Ordinary Skill in the Art.....	12
C.	“the first and second nodes are each coupled to a fault tolerant storage system (FTSS)”	13
D.	“highly reliable fault-tolerant storage media”	18
V.	The '738 Patent	21
A.	Overview.....	21
B.	Level of Ordinary Skill in the Art.....	26
C.	“context information”	27
D.	“potentially inaccurate or incomplete”	28
VI.	The '575 Patent.....	29
A.	Technical Background: Polymorphism.....	29
B.	Overview.....	36
C.	Level of Ordinary Skill in the Art.....	41
D.	“polymorphic metadata”	42
E.	“assembler”	44
F.	“disassembler”	45
G.	“a logic that...”	46
a.	Claim 6	48
b.	If Claim 6 were found to be means plus function, structure is nevertheless disclosed.....	49
c.	Claim 7	50

d.	If Claim 7 were found to be means plus function, structure is nevertheless disclosed.....	51
e.	Claim 11	52
f.	If Claim 11 were found to be means plus function, structure is nevertheless disclosed.....	53
g.	Claim 12	53
h.	If Claim 12 were found to be means plus function, structure is nevertheless disclosed.....	55
H.	“a business rules engine that...”	55
VII.	Conclusion	58

Table of Exhibits

Ex.	Document
1.	U.S. Patent No. 6,889,244 (“’244 Patent”)
2.	U.S. Patent No. 6,938,071 (“’071 Patent”)
3.	U.S. Patent No. 5,815,649 (“Utter”)
4.	U.S. Patent No. 6,574,655 (“Libert”)
5.	U.S. Patent No. 7,936,738 (“’738 Patent”)
6.	Oxford University Press, <i>A Dictionary of Computing</i> (5th Ed. 2004)
7.	U.S. Patent No. 7,313,575 (“’575 Patent”)
8.	Wikipedia, <i>Polymorphism (Computer Science)</i> , (Jun. 1, 2004)
9.	Barbara Liskov, <i>Program Development in Java</i> (2001)
10.	Douglass Downing and Michael Covington, <i>Dictionary of Computer and Internet Terms</i> , Seventh Ed. (2000)
11.	McGraw-Hill, <i>Dictionary of Electrical & Computer Engineering</i> (6th Ed. 2003)
12.	Steven M. Kaplan, Wiley Electrical Electronics Engineering Dictionary (2004)
13.	Thomas J. Sheffler, <i>A Portable MPI-Based Parallel Vector Template Library</i> (1995)
14.	Simon St. Laurent and Robert Biggar, <i>Inside XML DTDs</i> (1999)
15.	Microsoft Press, <i>Microsoft Computer Dictionary</i> (4th Ed., 1999)
16.	Business Wire, <i>Blaze Software Ships New Blaze Advisor Innovator, Letting Business Managers Adapt Systems to Changing Conditions without Programming</i> (June 6, 2000)
17.	Beth Bacheldor, <i>Personalized Shopping on the Internet</i> , Information Week (Oct. 4, 1999)
18.	Excerpts from the prosecution history of the ’244 patent

I. INTRODUCTION AND QUALIFICATIONS

1. My name is Dr. Martin Farach-Colton. I am the Leonard J. Shustek Professor of Computer Science and Chair of the Department of Computer Science and Engineering at New York University in New York, New York.
2. This report contains my opinions formulated to date and the reasons for those opinions. I may offer further opinions based on additional review of materials in this case, including opinions and/or testimony of other expert witnesses, within the guidelines of the Court.
3. In this section, I summarize my education, career history, publication, and other relevant qualifications.

A. Education

4. I received my Bachelor of Science degree in Mathematics and Chemistry from the University of South Carolina in 1984. I received my MD from The Johns Hopkins School of Medicine in 1988. I received my Ph.D. in Computer Science from the University of Maryland in 1991. My dissertation focused on string algorithms.

B. Career History

5. From September 1991 to August 1994, I was a postdoctoral fellow at DIMACS, the Center for Discrete Mathematics and Theoretical Computer Science at Rutgers University. My research there continued along the same lines as during my doctorate.
6. From September 1994 to December 2023, I was on the faculty in the Department of Computer Science at Rutgers University. When I left in 2023, I had been promoted to Distinguished Professor and am not Distinguished Professor Emeritus. While there, I worked on algorithms for large data, including streaming data, computational biology data, and the

design of databases and file systems. I taught courses on the design and analysis of algorithms.

7. During my time at Rutgers, I took various leaves of absence to work in industry. From July 1997 to August 1998, I was a Member of Technical Staff at Bell Labs. During that year, I worked on network algorithmics and well as on big data algorithms. From July 2000 to July 2002, I was a Senior Research scientist at Google, Inc. I worked on a variety of advanced algorithms for handling large data. I created several of the early features and products at Google. From January 2023 to June 2023, I was a Visiting Scientist at MIT.
8. In December of 2006, I founded Tokutek, a performance database company formed to commercialize my research. This was acquired in April of 2014. At the time of acquisition, clients included Walmart and Cisco.
9. From September of 2018 to March of 2024, I consulted for VMware Research Group. They implemented and deployed a key-value store of my design called SplinterDB.
10. Since January of 2024, I have been the Chair of the NYU Tandon Department of Computer Science and Engineering, and the Leonard J. Shustek Professor of Computer Science.
11. My current focus continues to be the design and analysis of algorithms, primarily for large data.

C. Publications and Patents

12. I have written approximately two hundred conference and journal papers, mostly on topics in the design and analysis of algorithms, or their implementation in storage systems such as databases and file systems. The full list is available on my CV.
13. I am a named inventor on the following patents:

- Bradley C. Kuszmaul, Michael A. Bender, Martin Farach-Colton, “Disk-resident Streaming Dictionary”, US Patent No. 8,489,638, July 16, 2023 – assigned to MIT, Stony Brook University and Rutgers University
- Michael A. Bender, Martin Farach-Colton, Yonatan R. Fogel, Zardosht Kasheff, Bradley C. Kuszmaul, Vincenzo Liberatore, Barry Perlman, Rich Prohaska, David S. Wells, “High-performance streaming dictionary”, US Patent No. 8,996,563, March 31, 2015 – assigned to Tokutek, Inc.
- Jeffrey A. Dean, Martin Farach-Colton, Sanjay Ghemawat, Benedict Gomes, Georges R. Hank, U.S. Patent No. 8,495,483, July 23, 2013 – assigned to Google, Inc.
- Martin Farach-Colton, Monika H. Henzinger, Bay Wei Chang, U.S. Patent No. 7,296,016, Nov. 13, 2007 – assigned to Google, Inc.

D. Other Relevant Qualifications

14. I serve as the Editor in Chief of ACM Transaction on Parallel Computing. I am also on the editorial board of Algorithmica. I have served as program committee chair of many conferences, including top conferences such as: Symposium on Discrete Algorithm and European Symposium on Algorithms.
15. I was a Sloan Fellow and have been named a Fellow of AAAS, ACM, IEEE and SIAM. I am a Member of the Argentine Academy of Sciences. I was the recipient of a Presidential/NSF grant Big Data.
16. I have taught Algorithms to thousands of students at Rutgers and received two teaching awards there.

E. Materials and Other Information Considered

17. I have considered information from various sources in forming my opinions, including the following:

- a) The Relevant Patents (*i.e.*, each of the '244, 738, and '575 patents);
- b) The prosecution histories of the Relevant Patents;
- c) Plaintiffs Valtrus Innovations Ltd. and Key Patent Innovations Ltd.'s P.R. 4-2 Disclosures dated November 29, 2024 and the extrinsic evidence identified therein for the Relevant Patents;
- d) Defendants' Preliminary Claim Constructions and Extrinsic Evidence dated November 29, 2024 and the extrinsic evidence identified therein;
- e) The materials cited in this Declaration, which are listed in the Table of Exhibits above.

18. I have also drawn on decades of experience in computer science research and industry, including my experience with large data algorithms and database system design.

F. Compensation

19. I am being compensated at my standard rate of \$750 per hour. No part of my compensation depends on the outcome of this case.

G. Prior Testimony

20. None.

II. SUMMARY OF OPINIONS

21. I have prepared this declaration at the request of the Plaintiffs in this case, Valtrus Innovations Ltd. and Key Patent Innovations Ltd. (collectively, "Valtrus"). This report

provides my opinions to date with respect to the parties' proposed constructions for certain claim terms of U.S. Patent Nos. 7,313,575 (the "'575 patent"); 7,936,738 (the "'738 patent"); and 6,889,244 (the "'244 patent") (collectively the "Relevant Patents").

22. I may provide further testimony and opinions if asked to testify in court or in a deposition. I also reserve the right to supplement or amend my opinions, as well as the bases for those opinions if asked to testify in court or in a deposition. I also reserve the right to supplement or amend my opinions, as well as the bases for those opinions, if I receive additional relevant information or to respond to opinions offered by other experts in this case.

III. UNDERSTANDING OF THE LAW

23. I am not a lawyer. My understanding of the relevant legal principles is based on information that was provided to me by counsel. I have applied the following legal principles provided to me by counsel in arriving at the opinions set forth herein.

A. Claim Construction

24. I understand that a patent may include two types of claims: independent claims and dependent claims. An independent claim stands alone and includes only the limitations it recites. A dependent claim can depend from an independent claim or another dependent claim and includes all the limitations that it recites in addition to all of the limitations recited in the claim or claims from which it depends.
25. I understand that a claim term or phrase of a patent is generally construed according to its plain and ordinary meaning as understood by a person of ordinary skill in the art ("POSITA") in the context of the patent at the time of the invention. I am informed that factors that may be considered in determining the level of skill include: the type of problems encountered in

the art; prior art solutions to those problems; rapidity with which innovations are made; sophistication of the technology; and education level of active workers in the field. If known, the education level of the inventor(s) may also be considered. A person of ordinary skill in the art is also a person of ordinary creativity, not an automation.

26. I understand that a claim term or phrase of a patent is construed in the context of the words of the patent claims, the specification of the patent (including figures or diagrams), and the prosecution history of the patent (including the cited references). This type of evidence is called “intrinsic” evidence. I understand that the context of any disputed claim term within the claim language itself is of primary importance. I further understand that it is improper to import limitations from the specification into a claim term or phrase through claim construction.

27. I understand that the Court may also look to additional evidence aside from intrinsic evidence, such as expert opinion (including opinions on what a POSITA would have known or understood at the time of invention), dictionaries, treatises, and testimony. This type of evidence is called “extrinsic” evidence. Extrinsic evidence can be useful to, for example, provide background on the technology at issue, explain how an invention works, ensure that the Court’s understanding of the technical aspects of the patent is consistent with that of a POSITA, or establish that a particular term in the patent or the prior art has a particular meaning in the pertinent field.

28. I understand that in certain instances a patentee may act as its own lexicographer so as to set forth a definition of a claim term other than its plain and ordinary meaning. However, I further understand that to do so it is not enough for a patentee to simply disclose a single

embodiment or use a word in the same manner in all embodiments, the patentee must “clearly express an intent” to redefine the term.

29. I understand that sometimes the ordinary meaning of claim terms as understood by a person of ordinary skill in the art may be readily apparent even to lay persons. I understand that claim construction in such instances involves little more than the application of the widely accepted meaning of commonly understood words.

B. Means-Plus-Function Claims

30. I understand that a claim element may be expressed as a means for performing a specified function without the recital of structure, material, or acts in support thereof, and that such elements are called “means-plus-function” terms.
31. I understand that claim terms that state the words “means for” performing a certain function are presumed to be means-plus-function terms.
32. I understand that claim terms omitting the phrase “means for” are presumed to not be means-plus-function terms. I further understand that when a disputed claim term omits the “means for” phrase the presumption can be overcome and means-plus-function interpretation will govern if the claim term fails to recite sufficiently definite structure or else recites function without reciting sufficient structure for performing that function. However, I understand that a claim term will not be interpreted as a means-plus-function term when that term recites structure that has a sufficiently definite meaning to a POSITA. I further understand that, in determining whether a claim term recites sufficiently definite structure, the term may be used in common parlance or by a POSITA to designate structure, even if the term covers a broad class of structures and even if the term identifies the structures by their function.

33. I understand that construing a means-plus-function term is a two-step process. First, one must identify the claimed function or functions of the term. Second, one must identify the corresponding structure in the specification, if any, that performs each function.
34. I further understand that, for the specification to disclose corresponding structure, the specification or prosecution history must clearly link or associate that structure to the function recited in the claim. I also understand that the disclosure in the specification must be of adequate corresponding structure to achieve the claimed function. The specification must disclose the structure such that a POSITA can recognize the structure in the specification and associate it with the corresponding function in the claim. I understand that it is insufficient merely for a POSITA to have been able to devise or otherwise implement a structure to perform the claimed function.
35. I understand that if a function recited by a means-plus-function limitation is computer-implemented—and a general-purpose computer cannot perform the function without special programming (software)—the specification must disclose the structure in the form of an algorithm. I understand that an algorithm is a step-by-step procedure for performing the claimed function. I understand that such an algorithm can be expressed in many forms, including flow charts, a series of specific steps, mathematical formula, prose, and so on. I understand that a restatement of the function itself in the specification is inadequate to disclose an algorithm for performing the claimed function.

C. Indefiniteness

36. I understand that a patent must include one or more claims that particularly point out and distinctly claim the subject matter regarded as the invention. I understand that Defendant

must show by clear and convincing evidence that a claim is indefinite. I understand that a patent claim is invalid for indefiniteness if, when read in light of the intrinsic evidence, the claim fails to inform, with reasonable certainty, those skilled in the art about the scope of the invention. I understand that absolute or mathematical precision is not required to comply with the definiteness requirement.

37. I understand that when a term of degree or subjective term is used in a claim, the patent must provide some objective boundary for a POSITA understand the scope of the invention.

38. I understand that a means-plus-function limitation is indefinite if a POSITA would be unable to recognize the structure in the specification and associate it with the corresponding claimed function. I understand that, if a means-plus function limitation is implemented by a general-purpose computer, the specification must disclose an algorithm for performing the claimed function.

IV. THE '244 PATENT

A. Overview

39. The '244 patent, entitled "Method and Apparatus for Passing Messages Using a Fault Tolerant Storage System," has a filing date of October 31, 2000.

40. Claim 1 of the '244 patent (the only independent claim) includes the terms "the first and second nodes are each coupled to a fault tolerant storage system (FTSS)" and "highly reliable fault-tolerant storage media" and states:

A method of transmitting messages between a first node and a second node, wherein the first and second nodes are each coupled to a fault tolerant storage system (FTSS), the method comprising:

transmitting a message from the first node to a communication agent in the FTSS;

storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS;

processing the message at the FTSS in accordance with a messaging paradigm;
and

transmitting the message from the FTSS to the second node.

41. The specification contextualizes its invention with reference to distributed applications and middleware systems:

In the art of computing, a distributed application is an application that is separated into two or more parts, such as a client application on a first computer system and a server application on a second computer system, with the two computers coupled together via a network. As used herein, the term “client” refers to an entity that requests a service, and the term “server” refers to the entity that fulfills the request. The entity can be an application, i.e., a client application or a server application, or the entity can be a computer system coupled to a network, i.e., a client computer system or a server computer system. Computer systems coupled to a network may also be referred to as “nodes”. The meaning of these terms will be apparent from the contexts in which the terms are used.

Id. at 1:27-40.

To ensure that all applications communicate seamlessly and reliably in a distributed application, the application designer may use a middleware product that distributes applications across multiple platforms, databases, and operating systems using message-based communications and, if desired, distributed transaction processing. Middleware products are used with client server applications to distribute processing among multiple servers, manage distributed transactions, and integrate multiple database platforms. Middleware systems are sometimes known in the art as “on-line transaction processing” (OLTP) systems. One such system is Bea Tuxedo®, which is a product of BEA Systems, Inc.

Id. at 2:4-16.

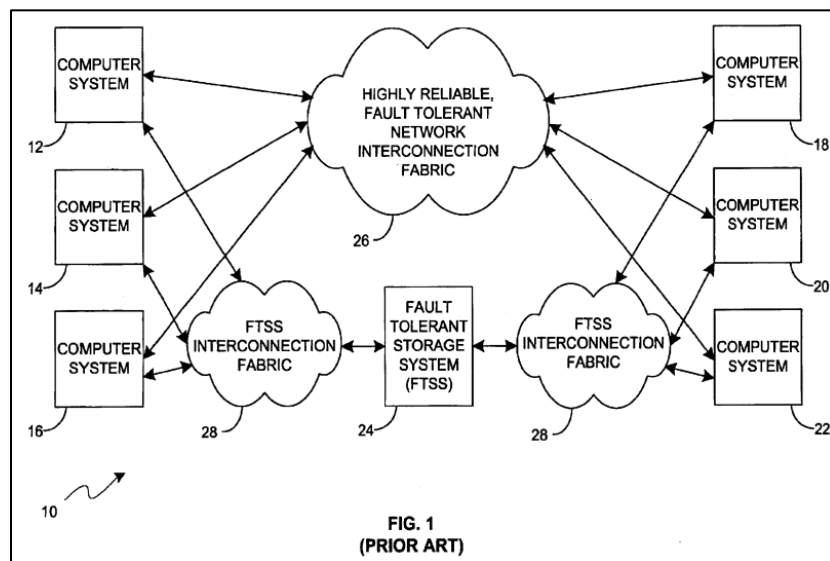
42. The specification refers to its invention as “a method and apparatus for passing messages between server and client applications using a fault tolerant storage system (FTSS).” *Id.* at 6:22-23. Separately, it notes that “The interconnection fabric that couples the FTSS to the

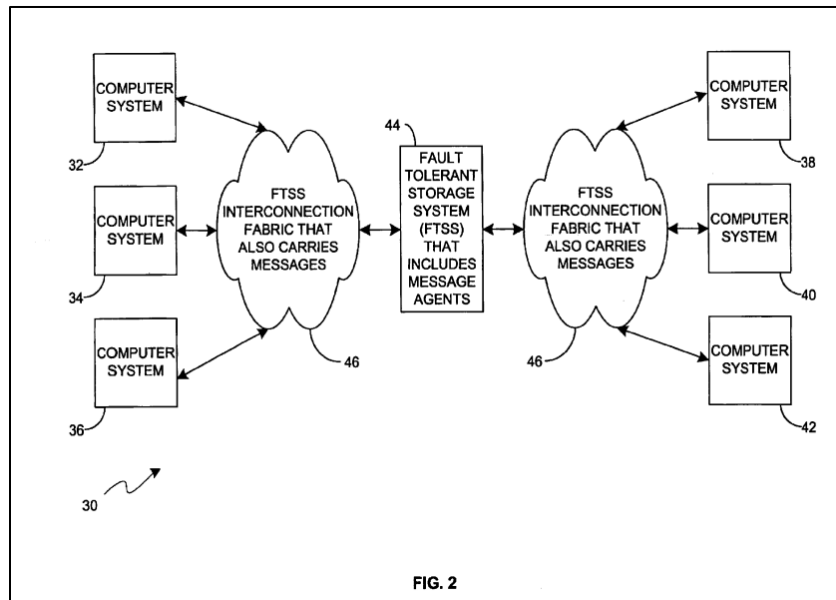
computer systems that host the client and server applications may also be used to carry messages.” *Id.* at 6:23-27.

43. The '244 patent describes its improvement to the prior art with references to Figs. 1 and 2.

Fig. 1 depicts a prior art system, whereas Fig. 2 “shows a highly reliable fault-tolerant system in accordance with the present invention that may be used to host a distributed application.”

Id. at 5:60-65.





44. Unlike the prior art Fig. 1, in which “fault tolerant network interconnection fabric 26 [carries] messages between the computer systems” (*id.* at 4:49-60), the invention uses a Fault Tolerant Storage System (“FTSS”) that “includes several message agents” such that the FTSS “also carries message traffic.” *Id.* at 5:17-21.

B. Level of Ordinary Skill in the Art

45. I understand that the relevant time period is around October 31, 2000, the filing date of the ’244 patent.

46. Based on my education and more than three decades of experience, as described above, I am familiar with the level of relevant knowledge about the patented technology of the ’244 patent.

47. In determining the level of skill of a person of ordinary skill in the art at the time of the invention, I have considered the factors outlined in Section III above.

48. It is my opinion that those of ordinary skill in the art during the relevant period would have had at least a B.S. in Computer Science or a related field, and 2-3 years’ experience working

with messaging and storage systems. However, I note that this description is approximate, and a higher level of education or skill might make up for less experience, and additional experience could make up for a lower education level.

C. “the first and second nodes are each coupled to a fault tolerant storage system (FTSS)”

49. I understand that Defendants contend that the plain and ordinary meaning of this term refers to “where the coupling to the FTSS provides for transmitting messages by a store-and-forward method (i.e., not through a conventional computer network such as a local area network (Ethernet) or wide area network (Internet)).” I disagree with Defendants’ contention.
50. In my opinion, “coupled” here would indicate to a POSITA simply that the first and second nodes are each communicatively connected to the fault tolerant storage system. Nothing in the plain and ordinary meaning of “the first and second nodes are each coupled to a fault tolerant storage system (FTSS)” would indicate to a POSITA that the communicative connection must exclude ethernet or internet.
51. This is supported by the disclosure of the ’244 patent. For example, the specification refers to the FTSS being “coupled to a server using a high-speed interface.” Numerous examples of “a high speed interface” are then given, including “an Ultra160 SCSI interface, a fiber channel interface, or an enterprise system connectivity (ESCON).” *Id.* at 4:35-48. These examples of networking technologies would indicate to a POSITA that numerous networking technologies can be used in the invention, including ethernet and internet.
52. This is further supported by the disclosure of ““A Fault Tolerant Storage System Having an Interconnection Fabric That Also Carries Network Traffic” by Blaine D. Gaither et al.,

change [*sic*] to Ser. No. 09/703,428” [the application leading to U.S. Patent No. 6,938,071 (the “’071 patent”)], incorporated by reference into the ’244 patent at 1:9-15. The ’071 patent refers to a node “coupled to [an] FTSS” via ethernet or the internet (directly contrary to Defendants’ construction):

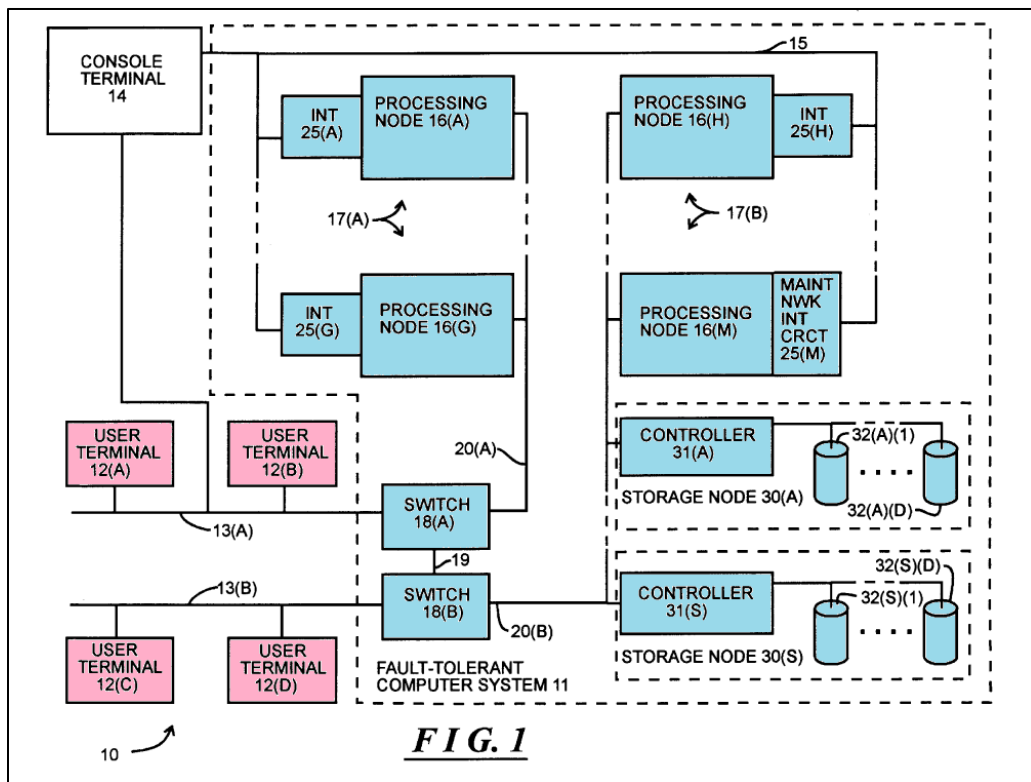
[T]he destination node is coupled to FTSS 92 via interconnection fabric 94 or external network 96.

...

Note that external network 96 is depicted in a generic manner. If external network 96 is an Ethernet network, then the packet is transmitted into the collision domain of the Ethernet network and is processed like any other packet transmitted by an external node coupled to the Ethernet network. Alternatively, external network 96 could represent a TCP/IP network such as the Internet, in which case the packet could be transmitted to a default gateway to begin the process of routing the packet to its final destination.

’071 patent at 11:43-12:5

53. The prosecution history of the ’244 patent is not inconsistent with the plain and ordinary meaning.
54. The issue of “conventional ethernet networks” came up in connection with a rejection the Examiner made over a reference to U.S. Patent No. 5,815,649 (“Utter”). July 18, 2003 Office Action (Ex. 18 at 3-4).
55. With respect to the limitation at issue “first and second nodes are each coupled to a fault tolerant storage system (FTSS),” the examiner cited to Figure 1 of Utter, which is reproduced below with color coding per the examiner’s apparent interpretation. Oct. 31, 2003 Office Action (Ex. 18 at 24-25). The user terminals 12A-12D (corresponding to the nodes) are highlighted in red, and the Fault-Tolerant Computer System (corresponding to the FTSS) is highlighted in blue. *Id.*; Utter at 2:61-3:15.



Utter at Fig. 1

56. The applicant explained the problems with the Examiner's rejection and how, in Utter, the communication between user terminals was conducted in a conventional manner, directly from one node to the other via Ethernet links 13(A) and 13(B), and "not through the FTSS." This can also be seen in connection with the above-pasted Figure 1, where the direct links between the user terminals (in red) are marked 13(A) and 13(B).

57. The relevant portion of the response is pasted below.

In section 4 of the Office Action, the Examiner states that "Utter discloses a method of transmitting messages between a first node and a second node, wherein the first and second nodes are each coupled to a fault tolerant storage system (FTSS)," referring to Figure 1 of Utter. Figure 1 of Utter shows a fault-tolerant computer system, the components of which are surrounded by an outer, dashed, nearly rectangular boundary shaped somewhat like a mirror-image of Utah turned on its side. Four user terminals (12(A)-(D)) and a console terminal (14) are connected to the fault-tolerant computer system via two *conventional* Ethernet networks (13(A)-(B)) (column 3, lines 59-61). Indeed, Applicants' representative agrees that, as shown in Figure 1, and according to Utter's disclosure, one of the four user terminals may exchange messages with another of the four user terminals via the two *conventional* Ethernet networks, and that the two user terminals are coupled through the two *conventional* Ethernet networks to a fault-tolerant computer system. However, in Utter's system, communications between user terminals is conducted in an entirely conventional manner, via entirely conventional Ethernet links, and not through the FTSS, as clearly claimed in the elements of claim 1 directed to Applicants' invention, and as discussed below.

April 30, 2004 Response to Office Action (Ex. 18 at 41).

58. In other words, the applicant was simply stating that the claimed invention required passing the message through the FTSS, and not directly from one node to another through networks 13(A) or 13(B). There is nothing that the applicant said in this section of the response pertaining to store and forward methods or disclaiming use of Ethernet networks or the Internet. The point was only that the message needed to pass *through the* FTSS during transmission from one node to the other, which the applicant argued was not disclosed in Utter.

59. The only place the applicant mentions store and forward is in the context of a different argument made later in the response, that also does not support SAP's construction. Further context is useful in explaining these statements.

60. In an earlier, November 24, 2003 Response to Office Action, the applicant stated that, "In order for a cited reference to anticipate claim 1, the cited reference would need to show

messages passed between a server computer and a client computer through an FTSS.” (Ex. 18 at 19).

61. In the April 30 Response to Office Action, the applicant explained that the passage of Utter cited by the examiner in response (reproduced below) does NOT disclose a message that is passed through the FTSS.

In addition to transferring requests from the user terminals 12(d) to the fault-tolerant computer system 11, the networks 13(n) may also download data from the fault-tolerant computer system 11 to the user terminals 12(d) for the local processing by the user terminals 12(d) or for display to a user on a display terminal which may be provided with respective ones of the user terminal 12(d). In addition, the networks 13(n) may transfer data from the user terminals 12(d) to the fault-tolerant computer system 11 for processing or storage, and in addition may transfer data among the user terminals 12(d). In one embodiment, the networks 13(A) and 13(B) are each in the form of *conventional* high-speed (100 Mb/second) Ethernet networks, which transfer information in the form of messages. As is conventional in Ethernet networks, messages generated by one device connected to a network 13(A), 13(B) (that is, by a user terminal 12(d) or by a fault-tolerant computer system 11) contain information to be transferred, as well as an address which identifies the intended recipient or recipients of the information. (emphasis added)

April 30, 2004 Response to Office Action (Ex. 18 at 44).

62. Per the applicant, this describes three situations: “(1) the networks can transfer data from the FTSS to a user terminal; (2) the networks can transfer data from user terminals to the FTSS; and (3) the networks can transfer data from one user terminal to another.” *Id.* However:

[T]here is no mention in the cited passage that the Ethernet networks would behave in a decidedly unconventional fashion by arranging for a message sent by a first user terminal to a second user terminal using the address of the second user terminal to not be directly sent from the first user terminal to the second user terminal, but to be instead intercepted by the FTSS, to which the message is not addressed, stored in highly reliable FTSS data storage, retrieved from the FTSS data storage, and then forwarded to the second computer. It is not surprising that the Examiner’s proposed interpretation of Utter is not disclosed or suggested in Utter, because it would not make sense to do so.

Id.

63. In other words, per the applicant, Utter disclosed that messages could be directly passed between one node and another (and not through the FTSS) using standard addressing

techniques, so there was no reason for a POSITA to think this involved re-routing those messages into the FTSS of Utter.

64. Ultimately, the applicant's only point (through both arguments) was that Utter did not disclose using the FTSS as the intermediary through which the messages were passed. As the applicant stated in the August 30, 2004 Appeal Brief, "[t]here is no teaching, disclosure, or even suggestion in Utter for two user terminals to exchange messages with one another by a mechanism in which the messages are stored in fault-tolerant storage media in one or more of the storage nodes 30(A)-30(S) within the fault-tolerant computer system." (Ex. 18 at 51).
65. For the reasons above, it is my opinion that a POSITA would have understood "the first and second nodes are each coupled to a fault tolerant storage system (FTSS)" to be construed under its plain and ordinary meaning, which is not limited to any "unconventional" computer network.

D. "highly reliable fault-tolerant storage media"

66. I understand that Defendants contend that this term is indefinite because "highly reliable" is a term of degree and fails to provide objective boundaries. I disagree with Defendants' contention.
67. The term "highly reliable fault tolerant storage media" as it appears in claim 1 ("storing the message in a data structure in highly reliable fault-tolerant storage media of the FTSS") would have been readily understood by a POSITA as referring to the physical storage media of the FTSS.

68. A POSITA would be able to identify “highly reliable fault tolerant” storage media. As used in the specification, “highly reliable” would not refer to any specific value or measurement of reliability, but rather as itself a quality that a system could provide.

69. This is consistent with the use of the term in the specification of the ’244 patent, which use “highly reliable” and “fault tolerant” to generally describe systems with additional indications or guarantees of reliability beyond non-redundant, single point of failure systems.

E.g.:

[Q]ueue-based communication is highly reliable and messages can even be delivered after failures, such as power outages or network interruptions.

Id. at 3:9-11.

Several interconnection fabrics are available to provide guaranteed delivery of packets. For example...ServerNet® hardware ensures data integrity using multiple techniques. For example, command link integrity isolates single-bit errors, cyclic redundancy checks maintain both data and control integrity of transfers crossing the interconnect fabric, and hardware protocol acknowledgments ensure that end-to-end reliable data transfer has occurred. In addition, ServerNet® is capable of providing a high level of availability by incorporating spare alternate paths within a primary interconnection fabric, and providing a fully redundant secondary interconnect fabric with redundant secondary attachments to network nodes. ...While interconnection fabrics such as ServerNet® and the Coupling Facility are fast, highly reliable, and fault tolerant, they are also very expensive.

Id. at 4:9-28.

Note that interconnection fabrics 26 and 28 are both highly reliable and fault tolerant. ... interconnection fabric 26 does not include any type of persistent storage. If a client suffers some type of error after receiving a message, the client is not able to request retransmission of the message from fabric 26. What is needed in the art is a practical way for a customer to use the same highly reliable and fault tolerant interconnection fabric to carry both file I/O transactions and messaging traffic, and also allowing a client to request retransmission of messages.

Id. at 4:62-5:6.

The highly reliable and fault tolerant nature of FTSS 44 ensures that FTSS 44 can guarantee delivery of a message transmitted from a sending computer system to a destination computer system. To see how this is accomplished, first consider a typical FTSS....The XP512 has a battery-protected fully redundant nonvolatile write cache that can range in size from 2 gigabytes to 32 gigabytes. As soon as a message is received by FTSS 44 from a sending computer system, the message is committed to the nonvolatile write cache. Thereafter, the message is written to a redundant array of independent disks (RAID) of FTSS 44, and processed by one of the message agents.

Id. at 6:54-6:67.

70. Additionally, the specification notes at 4:5-7 that “a designer of a distributed application will often specify that the application be implemented using highly reliable and fault tolerant interconnection fabrics and storage systems.” And the specification gives numerous examples of “highly reliable and fault tolerant” systems, *e.g.* SureStore® XP512 Disk Array; Coupling Facility provided by IBM Corporation; and ServerNet® from Compaq Computer Corporation. *Id.* at 4:29-48.

71. This is also consistent with how the term is used in the prior art cited on the face of the ’244 patent. For example, Utter describes the construction of “high-reliability” and “fault tolerant” computer systems:

It is generally possible to build computer systems which have extremely reliable components to accomplish tasks such as these and numerous others, and to provide preventive maintenance in such a way and with such frequency that failures are extremely improbable. However, such high-reliability computer systems would be extremely expensive to build and maintain. Accordingly, “fault-tolerant” computer systems have been developed, which is generally designed with the expectation that one or more element of the system may fail at some point in its operation, but that if an element does fail, other elements are available to detect the failure and ensure that the system will continue to give proper results.

Utter at 1:41-53.

72. And “high reliability” is also described as an attribute of television broadcast “devices” in

U.S. Patent No. 6,574,655 at 1:30-53:

One example is the modern television broadcast facility which has various types of components from different manufacturers that must be integrated to create an efficient, reliable environment for acquisition, production, transmission and archival of material....These devices are the workhorses of the station and are distinguished primarily by attributes of high availability, reliability and quality.

73. For the reasons above, it is my opinion that a POSITA would have understood “highly reliable fault-tolerant storage media” to be construed under its plain and ordinary meaning.

V. THE '738 PATENT

A. Overview

74. The '738 patent, entitled “Fault tolerant Systems,” has a filing date of July 6, 2004. The '738 patent is directed to “a method and system for providing fault tolerance in, for example, message-based communication systems.” *Id.* at 1:1-5.

75. The '738 patent explains that many critical systems, such as telecommunications networks, require high-availability, wherein acceptable downtime is on the order of a few minutes per year. Accordingly, many such systems are designed to be fault tolerant in a variety of ways.

As described by the specification of the '738 patent:

Different levels of high-availability exist which may be split into two broad categories, referred to herein as ‘service continuity’ and ‘task preservation’. Service continuity refers to the ability to continue to use the services provided by a system after a fault or switchover of a high-availability system, whereas task preservation, a higher level of high-availability, refers to the ability for tasks being processed when a fault or switchover occurs to be largely unaffected by the switchover.

Id. at 1:27-35

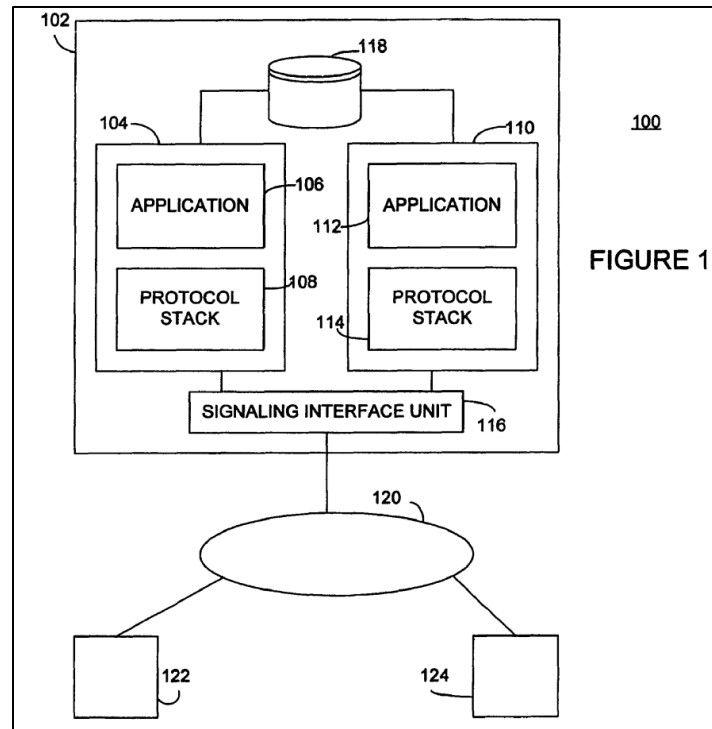
In order to provide task preservation a common storage element is often provided in addition to a high-availability configuration. Context data relating to each task

is stored in the common storage area, and may be used in the event of a switchover for reinitializing the new active system with the context of any tasks which were in progress when the switchover occurred.

Id. at 1:45-53

76. The '738 specification further explains that “the requirement for a common storage area for storing context data adds to the complexity and cost of such systems.” *Id.* at 1:63-65. As such, the '738 patent aims to describe a new fault-tolerant messaging architecture and configuration that reduces the cost and complexity associated with prior art systems that relied on common storage areas for high-availability systems. *Id.* at 1:66-3:45.

77. Fig. 1 depicts a prior art high availability system with two peer systems 104 and 110, which are arranged in a fault tolerant configuration “in which one of the systems is arranged in an ‘active’ mode, whilst the other system is arranged in a ‘standby’ mode. In the event of a fault being detected in the current active system, a switchover will occur such that the current active system becomes the standby system, and vice-versa.” *Id.* at 4:6-13. In the example of Fig. 1, the systems communicate with telephone switch 122 over network 120. *Id.* at 3:65-4:5. It relies upon a common storage area 118 for storing context data that can be used to rebuild context in the event of a switchover. *Id.* at 4:20-24.



Each of the peer systems 104 and 110 comprise numerous elements, such as an application (106 and 112 respectively) and a protocol stack (108 and 114 respectively). The systems 104 and 110 also have access to a common storage element, such as a database 118. As calls are processed on the active server the application 106 may decide to store context data in the storage element 118. The context data may relate, for each call, to both application context data and context data relating to one or more layers of the protocol stack 108. In the event of a switchover from the active server to the standby server, the stored context data may be retrieved by the application on the new active server, and be used to rebuild, for each call, the context of the application 112 and the protocol stack 114. This helps ensure that processing of calls open at the time of the fault is continued once the new active server has been initialized.

Id. at 4:18-33.

78. Fig. 4 shows a block diagram of a system according to an embodiment of the invention of the '738 patent:

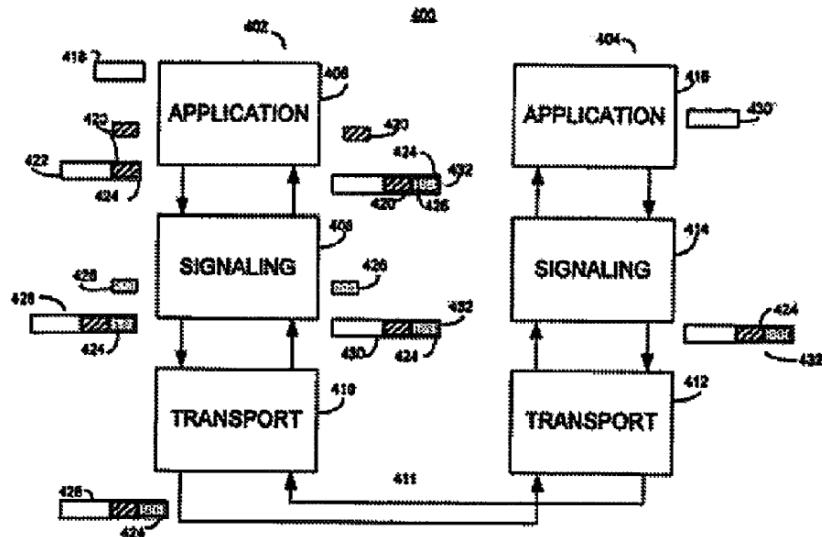


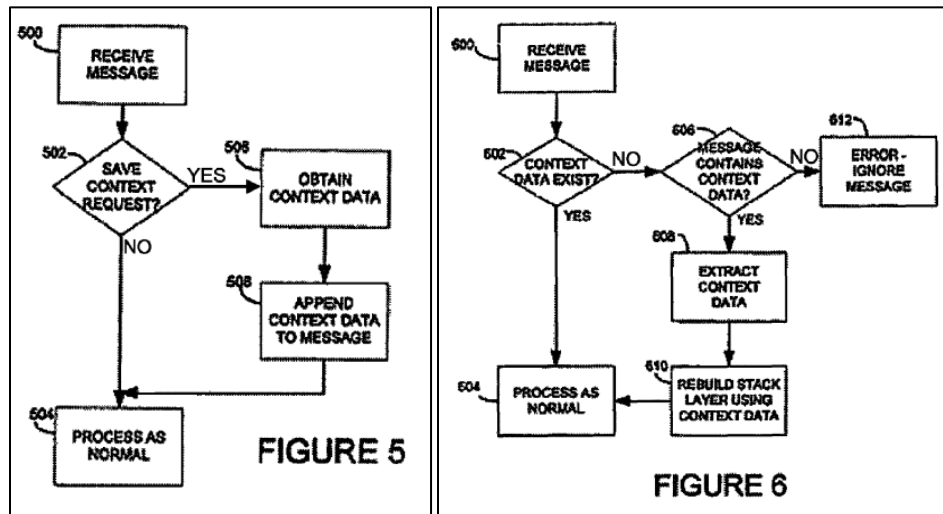
FIGURE 4

Referring now to FIG. 4, there is shown a block diagram of an exemplary system 400 according to an embodiment of the present invention, in which a HA node 402 communicates with a network node 404 across a network 411. Each of the nodes 402 and 404 comprise a protocol stack each having an application layer (406 and 416 respectively), a signaling layer (408 and 414 respectively) and a transport layer (410 and 412 respectively)....

An application (not shown) at the application layer 406 of HA node 402 sends an outgoing message 418 to an application at application layer 416 across the network 411. Additionally, the application indicates that context data should be stored to enable the context of one of more layers of the protocol stack of system 402 to be rebuilt in case of switchover the high-availability node 402. An application may indicate that it requests context data of the protocol stack to be stored in a number of different ways. For example, the application may add an additional field or flag when sending the message 418 via an API or function call used for sending messages. Alternatively, a flag may be set in the message sent from the application which indicates to each layer whether context data should be stored. It may also be desirable to automatically store context data for the protocol stack in every message sent from a HA node. In this case, it may not be required for the application to request that the context data should be stored, since this may be configured to occur automatically.

Id. at 5:12-42.

79. Figure 5 outlines a process by which a message is received at the different layers of the protocol stack and how context data is appended at each layer. Figure 6 outlines a process by which context data is received at the HA node and how the layers in the protocol stack can be rebuilt, if necessary, using that context data:



80. Accordingly, “the need for a high-availability system to store context data in a central data store is removed by appending context data to a message sent from a HA system, such that a message sent in response thereto contains the context data. In the event of a switchover, the context data received in a response message may be used to restore the context data of one or more layers of the protocol stack of the HA system....” *Id.* at 5:3-10.

81. Claims 1 and 33 of the '738 patent (the only asserted independent claims) state:

1. A method of storing context information in an outgoing message sent from a node including a computing device using a protocol stack having at least one layer, comprising:
providing, by the computing device, the outgoing message from an application to a layer of the protocol stack, the outgoing message is destined for an application on a destination node;
selectively indicating to the layer of the protocol stack that context information is to be obtained for that layer;
obtaining, by the computing device, context information in accordance with the indication; and
adding, by the computing device, the obtained context information to the outgoing message such that a response, received from the destination node, to the outgoing message contains the obtained context information.

33. A method of sending a message from a node through a hierarchical structure of one or more discreet layers comprising:
indicating to a layer that context information is to be obtained for that layer;
obtaining, by a computing device, context information in accordance with the indication; and
adding, by the computing device, the obtained context information to the message, such that a response to the message contains the obtained context information needed to restore a pre-switchover context of the layer.

B. Level of Ordinary Skill in the Art

82. I understand that the relevant time period is around July 6, 2004, the filing date of the '738 patent.

83. Based on my education and more than three decades of experience, as described above, I am familiar with the level of relevant knowledge about the patented technology of the '738 patent.

84. In determining the level of skill of a person of ordinary skill in the art at the time of the invention, I have considered the factors outlined in Section III, above.

85. It is my opinion that those of ordinary skill in the art during the relevant period would have had at least a B.S. in Computer Science or a related field, and 2-3 years' experience working with networking and fault tolerant systems. However, I note that this description is

approximate, and a higher level of education or skill might make up for less experience, and additional experience could make up for a lower education level.

C. “context information”

86. I understand that Defendants contend that this term is indefinite. I disagree with Defendants’ contention because a POSITA would have understood this term.

87. As used in the ’738 patent, “context information” (also referred to as “context data”) has about the same meaning as it would to a lay person, which I would describe as background or situational information related to a current activity, similar to state information. Context information is generally analogous to metadata. For example, emails will have subjects and bodies with content, but if the emails were lost and needed to be restored, additional information would be required, including metadata such as the time it was sent and received, the author, and the recipients. This is akin to “context information,” as it is used in the ’738 patent.

88. This is consistent with how the ’738 patent uses the term in relation to “a switchover of the active and standby systems occurs such that the standby system becomes the active system, and vice versa.” *Id.* at 1:21-23. “In the event of a switchover, the context data received in a response message may be used to restore the context data of one or more layers of the protocol stack of the HA [high availability] system.” *Id.* at 5:7-10. “[I]n order to store the context data representing the state of the protocol stack of the HA node, the HA node is required to include the context data in the TAG field of the INVITE message.” *Id.* at 7:22-25.

89. The '738 patent also explains that “context information” was commonly used in the prior art and “may relate to the state of different protocol layers of a protocol stack as well as any application specific data related to individual calls.” *Id.* at 1:53-56.
90. This understanding is also consistent with the specific examples given in the specification. *E.g.*, for “a billing application...required to know the time a call is established and the time a call ended in order to generate a billing record” context information can relate “to the start time of the call.” *Id.* at 7:33-35. And for SIP, “context data may include a destination address of a SIP call, which would enable SIP messages related to that call to be routed to the correct destination.” *Id.* at 7:1-3.
91. This understanding would be familiar to a POSITA. “Context switching,” referring to a similar process was a known term. For example, Oxford University Press, *A Dictionary of Computing, Fifth Edition* (2004) defines “context switch” as “when the environment for the currently active process is replaced by a new environment.” (Ex. 6 at -930).
92. For the reasons above, it is my opinion that a POSITA would have understood “context information” to be construed under its plain and ordinary meaning

D. “potentially inaccurate or incomplete”

93. I understand that Defendants contend that this term is indefinite. I disagree with Defendants’ contention.
94. The only asserted claim in which this term appears is claim 10, which states:

The method of claim 1, further comprising adding, to the message, an indication associated with the obtained context information where it is determined that the obtained context information is potentially inaccurate or incomplete.

95. This term (and the claim in which it occurs) is not particularly technical. It means what it says and would be readily understood by a POSITA just as well as it would by a lay person. If something is potentially inaccurate or incomplete, then it can be labeled “potentially inaccurate or incomplete.”

96. *E.g.*, in the billing application example, “an indeterminable amount of time may exist between the time the phone starts ringing and the time the call is answered.” Accordingly:

[B]y having to store context data relating to the start time of the call before the call is answered may lead to inaccurate context data being stored. In the event of a switchover occurring, and the protocol stack being re-initialized as described above, the restored billing information may be inaccurate, which could result in a user being overcharged.

Hence, preferably any billing record generated as a result of restored context data should be flagged as such, thereby allowing the service provider to either offer the call free of charge, or to reduce the cost of the call in question to take into account that the billing record generated thereby may be inaccurate. Alternatively, it may be preferable **to flag the context data as being inaccurate, or at least potentially inaccurate, when adding the context data to the outgoing message.**

’738 patent at 7:35-52.

97. For the reasons above, it is my opinion that a POSITA would have understood “potentially inaccurate or incomplete” to be construed under its plain and ordinary meaning.

VI. THE ’575 PATENT

A. Technical Background: Polymorphism

98. Polymorphism is an important concept in computer science. Literally, it refers to the ability of computational entities, such as functions or data, to take many forms. This is obvious from its Greek roots: “poly” means “many” and “morph” means “form.”

99. While the literal meaning is correct, implementations of Polymorphism are actually best understood as two related concepts: the ability for functions or procedures to be polymorphic with respect to the arguments they take and the ability for data structures to be polymorphic with respect to the objects they contain.

100. To provide an example, we can define a polymorphic function such as `print()`. As shown below, the `print()` function may take as an input various data types such as integers, decimal formatted numbers, or strings (i.e., sequences of characters), but the output will always be an appropriately formatted string containing a number with one decimal place. Necessarily, the exact algorithm implemented by calling the function (e.g., whether to convert to a string or add a decimal place) will change based on the type of value the function took as an input.

Function and Input	Output
<code>print(20)</code>	"20.0"
<code>print(20.0)</code>	"20.0"
<code>print("twenty")</code>	"20.0"
<code>print("1234")</code>	"1234.0"

101. A polymorphic data structure can also be illustrated with an example. As shown below, the variable `polymorphic_list` is a polymorphic data structure because it contains unspecified forms of data, including integer, decimal, and string values. In contrast, the variable `monomorphic_list` is not polymorphic; it contains only decimal values.

Variable	Value
<code>polymorphic_list</code>	[1, "twenty", 4.0]
<code>nonpolymorphic_list</code>	[0.1, 0.2, 0.3]

102. The two polymorphism concepts shown above would have been well known and easily accessible to a POSITA. For example, they were explained on the publicly available

Wikipedia page for “polymorphism” in June of 2004—just prior to the filing date of the ’575 patent (Ex. 8 at 1).



WIKIPEDIA
The Free Encyclopedia

WIKIPEDIA

Polymorphism (computer science)

This is an old revision of this page, as edited by RedWolf ([talk](#) | [contribs](#)) at **00:57, 4 June 2004** ([→Subtyping polymorphism: dab Java](#)). The **present address (URL)** is a permanent link to this revision, which may differ significantly from the current revision.

In computer science, **polymorphism** is the idea of allowing the same code to be used with different types, resulting in more general and abstract implementations.

The concept of polymorphism applies to functions as well as types:

- A function that can evaluate to and be applied to values of different types is known as a *polymorphic function*.
- A datatype that contains elements of an unspecified type is known as a *polymorphic datatype*.

103. More reputable sources would have reflected the same. For example, the Addison-Wesley textbook *Program Development in Java* from 2001 (shown below) explains Polymorphism in substantially the same way (Ex. 9 at -588):

Sidebar 8.1 Polymorphism

- Polymorphism generalizes abstractions so that they work for many types. It allows us to avoid having to redefine abstractions when we want to use them for more types; instead, a single abstraction becomes much more widely useful.
- A procedure or iterator can be polymorphic with respect to the types of one or more arguments. A data abstraction can be polymorphic with respect to the types of elements its objects contain.

104. In object-oriented programming languages, such as Java, polymorphism takes on special significance. As implied in their name, object-oriented programming languages are built

around structures called `objects`, which can contain both attributes (*e.g.*, data) and methods (*e.g.*, functions). These objects implement polymorphism using a mechanism known as **hierarchy** (also called “**hierarchal inheritance**”), which allows objects of different types within the same class to be treated as instances of a common superclass.

105. To illustrate how hierarchy in object-oriented programming languages implements polymorphism, I will show the creation of two objects in Java source code: a `Cat` named `garfieldObject` and a `Dog` named `scoobyDooObject`.

106. To start, we define a data type at the top of the hierarchy (a “superclass”) called `Pet`. This data type will be similar to the decimal and integer types described above, but it describes animals instead of numbers. The top of the hierarchy is most general. We will get more specific about the kinds of pets at lower levels of the hierarchy.

107. In real life, dogs and cats are subtypes of pets, so in our example, `Dog` and `Cat` are subtypes of `Pet`. Pets have names, and so our `Pet` superclass will have a constructor (a type of function for creating the class) that takes a name as an input. Pets also make sounds, but without knowing what kind of pet it is, we can’t know what sound it will make, so we use a placeholder.

```

1 // Superclass
2 class Pet {
3     // Attribute common to all pets
4     String name;
5
6     // Constructor which we will use create Pet Objects
7     Pet(String name) {
8         this.name = name;
9     }
10
11     // Method to display the pet's name
12     void displayName() {
13         System.out.println("This pet's name is " + name);
14     }
15
16     // Placeholder for sound method (to be overridden)
17     void sound() {
18         System.out.println("Some generic pet sound");
19     }
20 }
21

```

108. Now we will create two types of pets, a Dog and a Cat. While cats and dogs share some similarities, there are also differences, including their names and the sounds they make. Cats meow, and dogs bark; an unknown dog is a “Good Boy,” and an unknown cat is a “Good Kitty.” This is reflected in the example code below. Note that the function `super()` calls the constructor of the superclass, in this case the constructor of `Pet` which takes a name as an input.

```

22 // Dog class extending Pet
23 class Dog extends Pet {
24     // Default constructor setting a default dog name
25     Dog() {
26         super("Good Boy");
27     }
28
29     // Constructor for a specific dog name
30     Dog(String name) {
31         super(name);
32     }
33
34     // Overriding the sound method
35     @Override
36     void sound() {
37         System.out.println("Bark");
38     }
39 }
40
41 // Cat class extending Pet
42 class Cat extends Pet {
43     // Default constructor setting a default cat name
44     Cat() {
45         super("Good Kitty");
46     }
47
48     // Constructor for a specific cat name
49     Cat(String name) {
50         super(name);
51     }
52
53     // Overriding the sound method
54     @Override
55     void sound() {
56         System.out.println("Meow");
57     }
58 }
59

```

109. Now we can create our Cat named `garfieldObject` and Dog named `scoobyDooObject`.

```

60 // Main class
61 public class Main {
62     public static void main(String[] args) {
63         // Create a default Dog object
64         Dog defaultDog = new Dog();
65         defaultDog.sound(); // Outputs: Bark
66         defaultDog.displayName(); // Outputs: This pet's name is Good Boy
67
68         // Create a specific Dog object (Scooby Doo)
69         Dog scoobyDooObject = new Dog("Scooby Doo");
70         scoobyDooObject.sound(); // Outputs: Bark
71         scoobyDooObject.displayName(); // Outputs: This pet's name is Scooby Doo
72
73         // Create a default Cat object
74         Cat defaultCat = new Cat();
75         defaultCat.sound(); // Outputs: Meow
76         defaultCat.displayName(); // Outputs: This pet's name is Good Kitty
77
78         // Create a specific Cat object
79         Cat garfieldObject = new Cat("Garfield");
80         garfieldObject.sound(); // Outputs: Meow
81         garfieldObject.displayName(); // Outputs: This pet's name is Garfield
82     }
83 }

```

110. We can also see how these objects have implemented polymorphism using hierarchy: calling identically named functions (e.g., “. sound ()”) on these different objects results in different outputs. Specifically, `garfieldObject.sound()` returns a “Meow” and `scoobyDooObject.sound()` returns a “Bark.” Accordingly, the function “. sound ()” in our example is a polymorphic function.

111. This understanding of polymorphism was well known in the prior art. For example, the Addison-Wesley textbook *Program Development in Java* from 2001 (shown below) explains a structure similar to our example (Ex. 9 at -588).

In Java, polymorphism is expressed through hierarchy. Certain arguments are declared to belong to some supertype, and then the actual arguments can be objects belonging to subtypes of that type. That supertype is often `Object`. In this case, the polymorphic abstraction is limited to using `Object` methods such as `equals` on its parameters. Sometimes the polymorphic abstraction needs to use additional methods, however, and in that case the supertype is chosen to provide those methods.

112. While it was known that hierarchy was not the only way of implementing polymorphism, the popularity of Java and other object-oriented languages would have made hierarchy one of

the most recognized implementations of polymorphism at the time of the '575 patent. I have collected some references below to illustrate this:

polymorphism the use of different procedures, each with the same name, which are associated with different object types. For example, procedures named **draw** could be associated with the types **point**, **circle**, and **square**. Calling **draw** for any particular object then activates the right drawing procedure for that type. See OBJECT-ORIENTED PROGRAMMING.

Douglass Downing and Michael Covington, *Dictionary of Computer and Internet Terms*, Seventh Ed. (2000) (Ex. 10 at -634).

polymorphism [COMPUT SCI] A property of object-oriented programming that allows many different types of objects to be treated in a uniform manner by invoking the same operation on each object.
{ ,pāl-i'môr,fiz-əm }

McGraw-Hill, *Dictionary of Electrical & Computer Engineering* (2003) (Ex. 11 at -858).

polymorphism 1. The occurrence of different forms or shapes. 2. In object-oriented programming, the ability to behave differently, or elicit a different response, depending on the circumstances. For example, a routine may be handled in a given manner depending on the characteristics of an object.

Steven M. Kaplan, *Wiley Electrical Electronics Engineering Dictionary* (2004) (Ex. 12 at -078).

There is a link between collection-oriented programming and object-oriented systems that is often not recognized. Most object-oriented systems provide polymorphic collection classes that manage heterogeneous sets of objects [4]. Traditionally, polymorphism is implemented through a class hierarchy using inheritance (although other mechanisms are possible). Common functionality is provided for a group of types by "inheriting" the functionality from a base class. All classes derived from this base class then have a minimal set of member functions that a collection class can use to manage its elements.

Thomas J. Sheffler, *A Portable MPI-Based Parallel Vector Template Library* (1995) (Ex. 13 at -946).

B. Overview

113. The '575 patent, entitled "Data Services Handler", has a filing date of June 14, 2004. The '575 patent is directed to "A data services handler comprises an interface between a data store and applications that supply and consume data, and a real time information director (RTID) that transforms data under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data." '575 patent at Abstract.
114. Claim 1 of the '575 patent includes the terms "polymorphic metadata," "assembler," and "disassembler" and states:

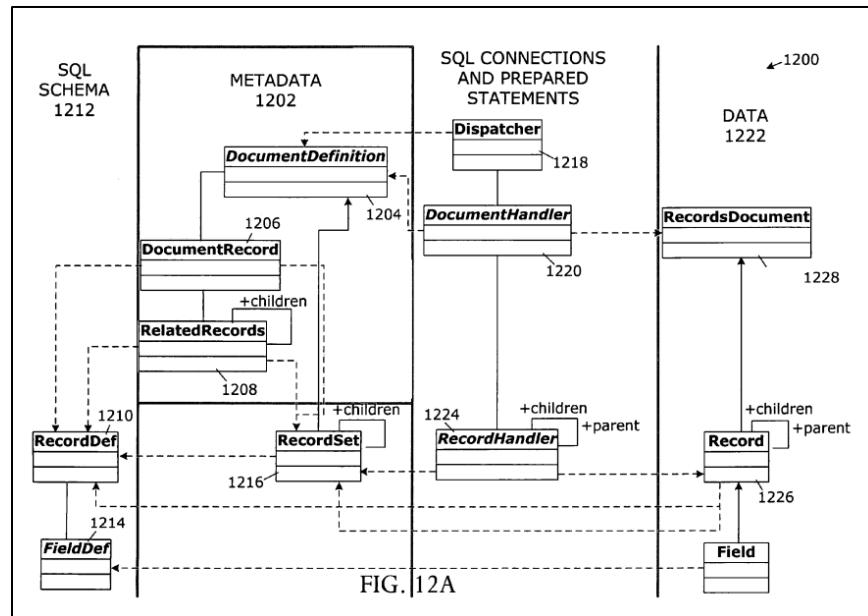
1. A data services handler for execution on a computing system comprising: an interface executing on the computing system for communicating between a data store and applications that supply and consume data; and a real time information director (RTID) executing on the computing system that transforms data for supply, consumption, or both by the applications under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data, the RTID to dynamically instantiate, create, and cache for reuse disassemblers and assemblers according to demand, the RTID further comprising: at least one disassembler defined according to the polymorphic metadata that disassembles inbound data documents into multiple records in preparation for insertion into metadata tables; and at least one assembler defined according to the polymorphic metadata that assembles outbound data documents from at least one record selected from at least one metadata-selected table whereby data is transformed for supply, consumption or both.	20 25 30 35
---	----------------------

115. As described in the specification, "a data services handler" comprises an interface between a data store and applications that supply and consume data, as well as a "real time information director (RTID)" which transforms data "under direction of polymorphic metadata that defines a security model and data integrity rules for application to the data." *Id.*, 5:6-14.
116. The RTID is described as an "integrated set of services." *Id.*, 3:22-24. Services provided by the RTID could include "security, auditing, data cleansing, deduplication, aggregation,

real-time scoring, business rules, accumulators, user-written functions, and the like.” *Id.*, 4:12-17. But chiefly, the “real time information director is responsible for most or all data input and output to the real-time data store.” *Id.*, 3:24-31. To do so, real time information director “prepares and caches Java Database Connectivity (JDBC) statements.” *Id.*, 3:46-48.

117. As described in the specification, “[a] ‘document disassembler’ contains prepared statements for inserting or updating various records related to a particular class of inbound documents into the data store. A ‘document assembler’ contains prepared statements for selecting various records related to a particular class of outbound documents from the data store. The real time information director can instantiate and cache for reuse as many disassemblers and assemblers of each document class as is appropriate to handle peak concurrent demand.” *Id.*, 3:48-57.

118. Figs. 12A through 12E “are Unified Modeling Language Diagrams [“UML”] describing the operations and functionality of the real time information director.” *Id.*, 2:34-36. As shown in, e.g. Fig. 12A below, “[d]ocument definitions determine behaviors of the real time information director.” *Id.*, 29:53-61. Note that in UML diagrams such as Figs. 12A through 12E, solid lines indicate association (e.g., one object or class contains the other) whereas dashed lines indicate dependency (e.g., one object or class uses the other).

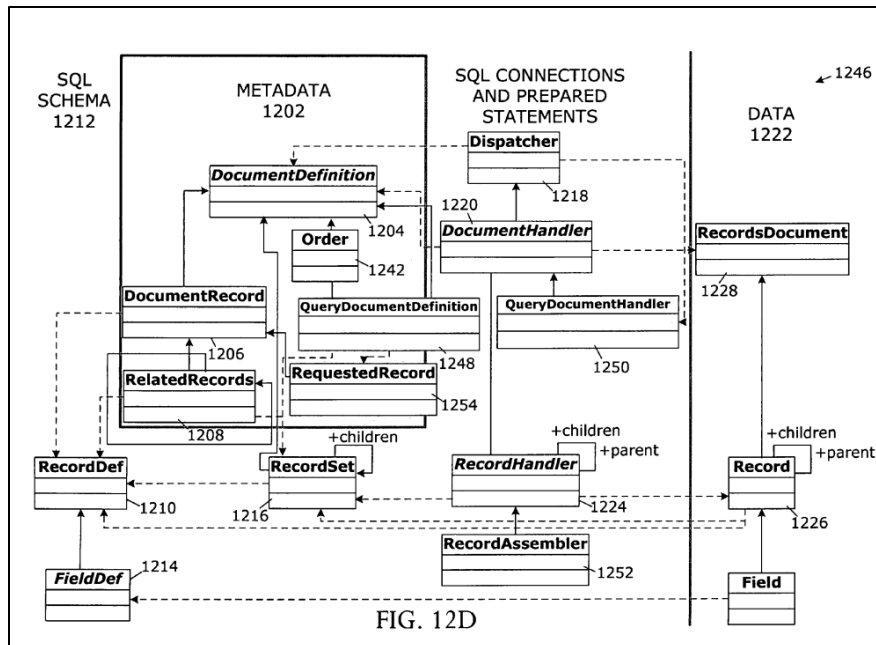


119. As shown and described, the Dispatcher 1218 of the RTID allocates and manages DocumentHandlers 1220 to process inbound documents and subscriptions. *Id.*, 30:22-34. The DocumentHandlers 1220 are created by the Dispatcher according to the DocumentDefinition 1204 metadata 1202. *Id.*, 29:58-67; 30:59-67.

The DocumentHandler 1220 includes a tree of RecordHandlers 1224. Each RecordHandler 1224 is constructed and prepares applicable Java DataBase Connectivity (JDBC) statements based on information in the RecordSet 1216 and associated RecordDef 1210. RecordHandlers 1224 either act on the appropriate Records 1226 in the inbound document to update the database, or construct the appropriate Records 1226 for the outbound Document 1228 after reading the database.

Id., 30:25-33.

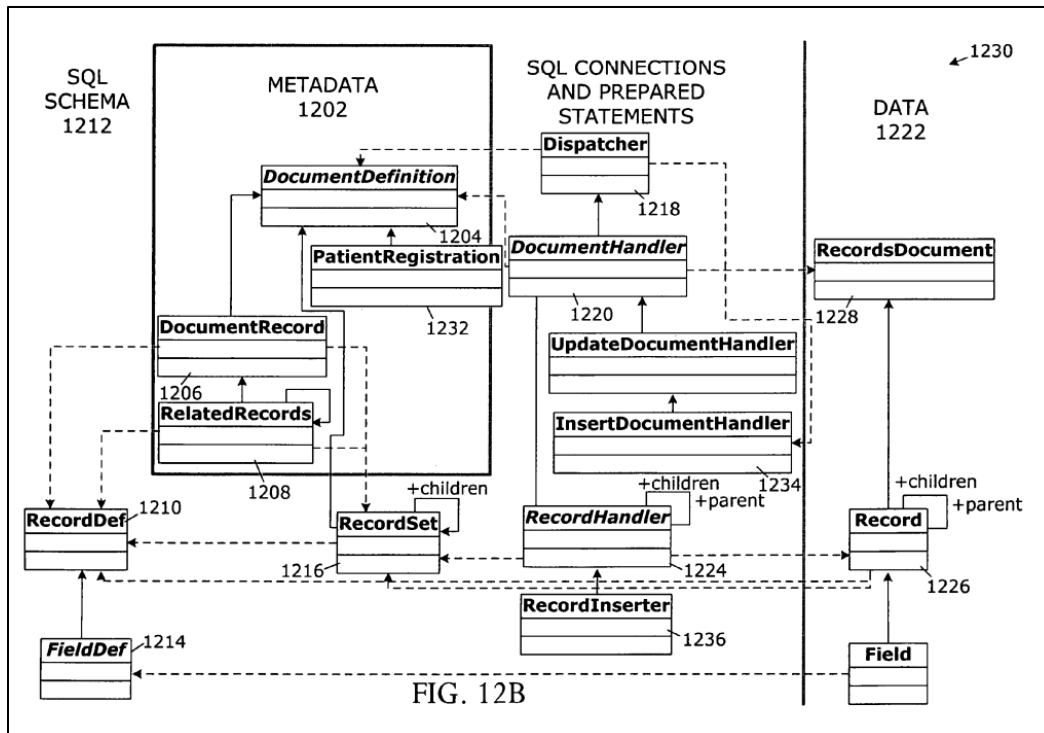
120. “One tree of RecordHandlers 1224, containing prepared statements, is constructed per database connection and DocumentHandler 1220. One tree of Records 1226 is constructed per Document 1228.” *Id.*, 30:36-44.
121. More detail on the structure of the RecordHandlers 1224 is shown in Fig. 12D, a UML diagram that illustrates the RTID when used to query a document. *Id.*, 31:48-50.



In the example, the inbound document is an order query. The top tag in the document is ORDER_R. The Dispatcher 1218 obtains a QueryDocumentHandler 1250 corresponding to this document. However, the QueryDocumentHandler 1250 is not created from the QueryDocumentDefinition 1248, but from the Order document definition 1242. The QueryDocumentHandler 1250 creates a RecordAssembler 1252 which creates child RecordAssemblers, grandchildren, and the like. The RecordAssembler 1252 contains a prepared select statement with clauses determined by the enriched RecordDef 1210.

Id., 32:3-14.

122. The RTID enables not just document queries, but the insertion of documents. Fig. 12B is a UML diagram of an Insert function. *Id.*, 30:45-50. The Dispatcher 1218 (using its child objects) parses and disassembles the inbound document into data 1222. *Id.*, 30:51-67.



In the illustrative example, a new patient is registered. A PatientRegistration 1232 document definition is self-identified as an insert document. The Dispatcher 1218 accordingly creates an InsertDocumentHandler 1234 for the PatientRegistration 1232. The InsertDocumentHandler 1234 creates a RecordInserter 1236 for the table named in the DocumentRecord 1206. That RecordInserter 1236 creates RecordInserters for each RelatedRecords 1208 that is child to the DocumentRecord 1206. Each RecordInserter 1236 in turn creates RecordInserters for any RelatedRecords 1208 that is child to the corresponding RelatedRecords 1208. Each RecordInserter 1236 holds the prepared statement to insert records into a particular table. The complete prepared statement is determined by the RecordDef 1210 for that RecordSets 1216.

Id., 31:1-14.

C. Level of Ordinary Skill in the Art

123. I understand that the relevant time period is around June 14, 2004, the filing date of the '575 patent.

124. Based on my education and more than three decades of experience, as described above, I am familiar with the level of relevant knowledge about the patented technology of the '575 patent.

125. In determining the level of skill of a person of ordinary skill in the art at the time of the invention, I have considered the factors outlined in Section III, above.

126. It is my opinion that those of ordinary skill in the art during the relevant period would have had at least a B.S. in Computer Science or a related field, and 2-3 years' experience working with databases or related information systems. However, I note that this description is approximate, and a higher level of education or skill might make up for less experience, and additional experience could make up for a lower education level.

D. "polymorphic metadata"

127. I understand that Defendants contend that this term is indefinite. I disagree with Defendants' contention.

128. Metadata would have been well understood as, plainly, data about data (*i.e.*, "metadata, defined as data that describes various aspects of other data" per the '575 patent at 4:18-19).

129. As I described above, polymorphism, polymorphic datatypes, and polymorphic functions all would have also been well understood by a POSITA. "Polymorphic metadata" as used in the claim would, accordingly, have reasonably informed a POSITA about the scope of the invention.

130. In my opinion, a POSITA would have understood "polymorphic metadata" to mean basically what it says: metadata defined through a polymorphic mechanism, such as hierarchal inheritance.

131. This understanding matches the disclosure in the '575 patent, which describes an embodiment of polymorphic metadata using hierarchy to implement the polymorphism. Indeed, the specification uses Java source code, just as I did in the example implementing polymorphism above. *E.g.*:

Polymorphic metadata can be used to enable extension of the real time information director to facilitate and address new functionality and behaviors. Polymorphic metadata is metadata that takes many forms, using polymorphism to flexibly represent different patterns economically and facilitate evolution of metadata as new circumstances arise. Generally, an individual metadata class can be used to represent a different common pattern, defined with attributes that are specific to the pattern. For example, circles and rectangles can be assigned to different classes of Shape with Circle having a defining attribute called diameter, and Rectangle having defining attributes called length and width. The polymorphic representation is economical because specified attributes or parameters of metadata can be limited to those that are applicable to a specific pattern under consideration. New metadata classes can be rapidly generated to solve new problems outside of the patterns previously encountered.

'575 patent at 4:34-50.

In some embodiments, polymorphic metadata can be expressed in Java, enabling usage of descriptive language for new classes of metadata to be implemented at the same time as the behavior that carries out that metadata.... In a Java implementation, individual classes of real time information director (RTID) metadata can be implemented as distinct Java classes. Parameters passed to the constructors for each metadata class serve to describe the metadata. Other methods of metadata classes are called to carry out the functions of the metadata. A new class of metadata is most frequently created by creating a new subclass of an existing metadata class. The subclass can be used in any context in which the superclass could be used. Additional parameters can be defined for the sub-class, which can then be used within the implementation of any methods that override the inherited behaviors of the superclass. The metadata classes are stateless and are called by the core functionality of the real time information director (RTID) whenever the real time information director is operating on relevant data. Stateful data objects are passed to the relative metadata methods as parameters.

'575 patent at 4:51-5:6.

Polymorphism enables different metadata objects to have different numbers and types of attributes, determined by class. Behaviors can be defined that are specific

to certain classes of objects, and the attributes specific to those behaviors need only be specified in those classes of objects, an approach that facilitates introduction of new behaviors and special cases, without modifying the metadata for non applicable or irrelevant objects and classes.

The metadata objects for the real time information director 302 can be constructed in the Java programming language. Individual classes of metadata can be embodied in a distinct Java class. Special cases of metadata are frequently represented as subclasses of more general metadata classes. Complex metadata can be represented as composite objects constructed from several metadata classes.

'575 patent at 13:5-19.

132. For the reasons above, it is my opinion that a POSITA would have understood “polymorphic metadata” to be construed under its plain and ordinary meaning.

E. “assembler”

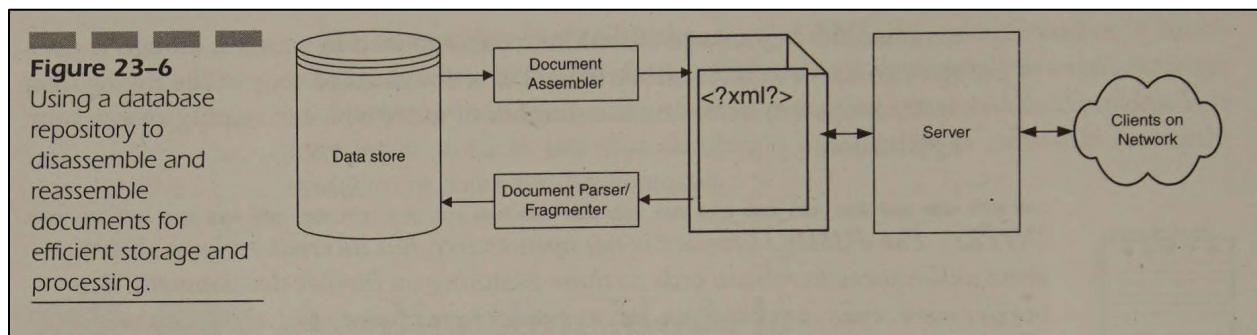
133. I understand that Defendants contend that this term is indefinite. I disagree with Defendants’ contention and agree with Plaintiffs that this term should be construed under the term’s plain and ordinary meaning.

134. The term assembler would have been well known to a POSITA, not just in referring to databases, but more generally referring to the software or instructions for converting from a system format to a target format. In the context of databases (and the '575 patent, of course), this would mean the instructions or software for getting the information out of the database and into a format to be used.

135. This is how the term “assembler” is explained in the '575 patent’s specification. “A ‘document assembler’ contains prepared statements for selecting various records related to a particular class of outbound documents from the data store. The real time information

director can instantiate and cache for reuse as many disassemblers and assemblers of each document class as is appropriate to handle peak concurrent demand.” *Id.*, 3:51-54.

136. This is consistent with how the term was used in the prior art. For example, Simon St. Laurent and Robert Biggar’s *Inside XML DTDs* from 1999 uses the term “assembler” when describing how “XML and relational databases” interact. Ex. 14 at -931. “Instead of the traditional file system, all these systems use structured back-ends to hold fragmented interpretations of XML documents, which can be reassembled as needed, as shown in Figure 23-6.” *Id.*



Simon St. Laurent and Robert Biggar, *Inside XML DTDs* (1999) (Ex. 14 at -932).

137. For the reasons above, it is my opinion that a POSITA would have understood “assembler” to be construed under its plain and ordinary meaning.

F. “disassembler”

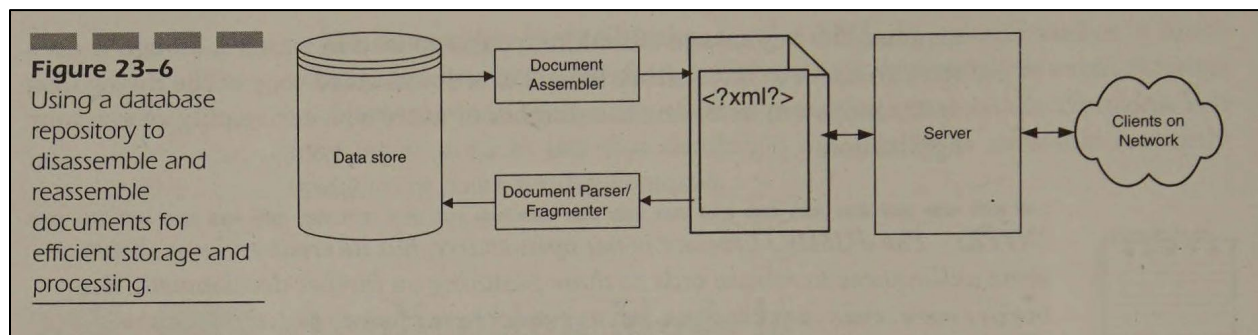
138. I understand that Defendants contend that this term is indefinite. I disagree with Defendants’ contention and agree with Plaintiffs that this term should be construed under the term’s plain and ordinary meaning.

139. The term disassembler is similar (and complementary) to the term “assembler.” It too would have been well known to a POSITA, again not just in referring to databases, but more generally referring to the software or instructions for converting from a source format into a

target format. In the context of databases (and the '575 patent), this would then mean the instructions or software for putting information into the database.

140. This is how the term “disassembler” is used in the '575 patent's specification. “A ‘document disassembler’ contains prepared statements for inserting or updating various records related to a particular class of inbound documents into the data store.... The real time information director can instantiate and cache for reuse as many disassemblers and assemblers of each document class as is appropriate to handle peak concurrent demand.” *Id.*, 3:48-57.

141. And this is how the term was used in the prior art. Again as an example, Laurent and Robert Biggar's *Inside XML DTDs* from 1999 uses the term “disassemble” to describe the functions of a “Document Parser/Fragmenter” when taking an XML document and inserting its information into a database.



Simon St. Laurent and Robert Biggar, *Inside XML DTDs* (1999) (Ex. 14 at -932).

142. For the reasons above, it is my opinion that a POSITA would have understood “disassembler” to be construed under its plain and ordinary meaning.

G. “a logic that...”

143. I understand that Defendants contend that the terms “a logic that...” in claims 6, 7, 11, and 12 should be interpreted as means-plus-function terms. I disagree.

144. In the context of the claims, a person of ordinary skill would have understood most of these “logic” terms to refer to features of the real time information director (RTID) software, the RTID itself being a software component of the broader Data Services Handler. This is evident from the claim language: each of claims 6, 7, and 11 requires “The data services handler according to claim 1 wherein the real time information director further comprises: a logic...” with claim 1 itself requiring that the data services handler comprise “a real time information director (RTID) *executing on the computing system.*” Emphasis added. A person of ordinary skill would have been familiar with this way of referring to software as “logic” and would have interpreted the claims accordingly for all such terms.

145. While claim 12, does not claim features of the RTID, it nevertheless claims features of the “data services handler,” which must comprise a “servlet” for invoking the RTID synchronously and “a logic” for invoking the RTID asynchronously. A POSITA would understand that this refers to the software on the data services handler, a specific structure for performing claimed functionality.

146. I note also that (while it does not contain the term “a logic that”) claim 35 is structured similarly to claims 6, 7, and 11, depending from claim 33 which depends from claim 1:

35. The data services handler according to claim 33 wherein the real time information director further comprises:

a query logic that queries the database to determine information about tables in the database that the document uses, the information including field name, field datatype, and field allowance of nulls.

A person of ordinary skill would have therefore understood that “a query logic” refers to software of the real time information director (RTID) executing on the Data Services Handler.

a. Claim 6

147. Claim 6 recites:

The data services handler according to claim 1 wherein the real time information director further comprises:

a logic that represents metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class.

148. Far from purely claiming a function, a POSITA would understand claim 6 is requiring specific features of the software behind the RTID. Similar to claim 1's requirement of polymorphic metadata, claim 6 requires metadata to be represented as polymorphic objects with "distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class," similar to the example of polymorphism implemented via hierarchy that I explained above using Java code.

149. This is consistent with the specification, which discloses that "the real time information director 302 represents metadata as polymorphic objects" that could be "constructed in the Java programming language."

Polymorphism enables different metadata objects to have different numbers and types of attributes, determined by class. Behaviors can be defined that are specific to certain classes of objects, and the attributes specific to those behaviors need only be specified in those classes of objects, an approach that facilitates introduction of new behaviors and special cases, without modifying the metadata for non-applicable or irrelevant objects and classes.

The metadata objects for the real time information director 302 can be constructed in the Java programming language. Individual classes of metadata can be embodied in a distinct Java class. Special cases of metadata are frequently represented as subclasses of more general metadata classes. Complex metadata can be represented as composite objects constructed from several metadata classes.

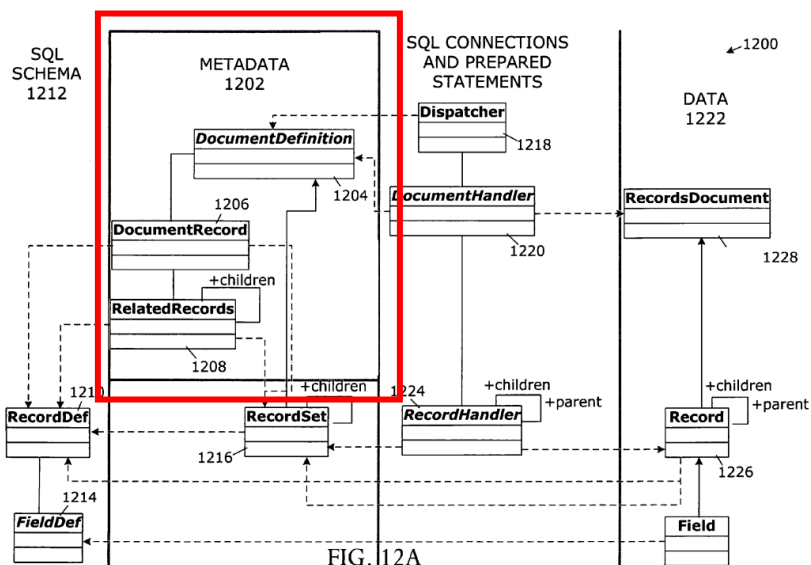
Id., 13:1-12.

150. A POSITA would have been familiar with Java and the use of Java classes sufficient to implement claim 6.

b. If Claim 6 were found to be means plus function, structure is nevertheless disclosed.

151. I have also been asked to consider what structure, if any, in the specification would correspond to claim 6's "a logic that represents metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class" should it be found to be a means plus function term.

152. A POSITA would understand that structure in the '575 patent corresponding to this term can be found in the depiction of the RTID metadata 1202 in Figs. 12A-12E (as I described above in Section VI.B), and the description of this metadata found throughout the patent, including in 29:62-36:47. For example, DocumentDefinition, DocumentRecord, and RelatedRecords are "individual metadata objects" with "distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class" as shown in Fig. 12A (outlined in red below).



Content metadata 1202 for a Document Definition 1204 is defined by a DocumentRecord 1206, which contains various RelatedRecords 1208. RelatedRecords 1208 may contain other RelatedRecords 1208.... One tree of RelatedRecords 1208, with DocumentRecord 1206 as the trunk, is constructed for each base document definition.

Id. at 29:62-66; 30:35-37.

153. Additionally, a POSITA would understand that hierarchy (or hierarchal inheritance) would describe a mechanism for implementing “metadata as polymorphic objects, enabling individual metadata objects to have distinct numbers and types of attributes according to a predetermined object class, the behaviors and attributes being defined specific to object class.” This is described at 3:65-4:11; 4:34-5:6; 13:1-36; 14:1-37; 15:30-35; and 16:13-17.

c. Claim 7

154. Claim 7 recites:

The data services handler according to claim 1 wherein the real time information director further comprises:

a logic that represents metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful metadata classes delegate operation to the stateless polymorphic metadata objects.

155. Similar to claim 6, a POSITA would understand claim 7 is requiring specific features of the software behind the RTID, not functionality to be performed by a generic computer.

Similar to claim 1's requirement of polymorphic metadata, claim 7 requires metadata be represented as stateless polymorphic objects.

156. Per the '575 patent:

As a general rule in object-oriented design, one strives to embed stateful data in classes containing the methods that operate on the stateful data. In contrast, in the real time information director 302, methods of stateful classes generally delegate operation to stateless metadata objects.

Metadata objects are far less numerous, are subject to much less frequent modification, and are much more frequently used, than individual data objects. Significant performance advantages can be attained by caching metadata objects for reuse. Performance can also be improved by sharing metadata objects between threads. Optimizations can be attained when the metadata objects are stateless. When metadata is modified, obsolete metadata can be discarded and new metadata instantiated.

Id., 13:37-50.

157. A POSITA would have been familiar with object-oriented design and Java classes, sufficient to implement claim 7.

d. If Claim 7 were found to be means plus function, structure is nevertheless disclosed.

158. I have also been asked to consider what structure, if any, in the specification would correspond to claim 11's "a logic that represents metadata as stateless polymorphic objects that are permanent after initial construction, whereby methods of stateful metadata classes

delegate operation to the stateless polymorphic metadata objects” should it be found to be a means plus function term.

159. A POSITA would understand that an algorithm or mechanism for implementing this term is described at 3:65-4:11; 4:34-5:6, as well as 13:20-27 (below).

In an illustrative system, metadata objects are stateless. The metadata objects are not modified after initial construction, a characteristic attained, for example, in the Java language by declaring all attributes of metadata classes as final. Such attributes are set only during the construction of the metadata object. The attributes are specified as parameters of the metadata object, or are constructed entirely from such parameters and other information available during the construction of the metadata object.

e. Claim 11

160. Claim 11 recites:

11. The data services handler according to claim 1 wherein the real time information director further comprises:

a logic that reads a schema of tables identified in metadata; and

a logic that automatically generates and caches prepare[d] SQL statements from a content metadata portion of document metadata according to the schema.

161. Claim 11 again requires specific features of the software behind the RTID. While unlike claims 6 and 7 it pertains to specific functions of the RTID, the functions must nevertheless be performed by the software of the RTID.

162. This is consistent with the disclosure of the specification:

The real time information director 302 automatically generates and caches prepared SQL statements from the content metadata after reading the schema of the tables identified in the metadata to obtain detailed information about names and data-types of individual columns. Parameterized select, insert, delete, and update statements are generated according to application to enable the real time information director 302 to manage inbound and outbound documents.

Id., 16:43-51.

Before inserting in the database, the real time information director looks up each column in the owning RelatedRecords, then the parent, then the grandparent, all the way up to the DocumentRecord. Columns can be moved between different tables, or completely removed, without changing the mapping metadata. Accordingly, mapping is equally driven by the static mapping metadata and the dynamically-read schema of the database.

Id., 51:50-57.

163. A POSITA would have been familiar with SQL and the reading of tables sufficient to implement claim 11.

f. If Claim 11 were found to be means plus function, structure is nevertheless disclosed.

164. I have also been asked to consider what structure, if any, in the specification would correspond to claim 11's "a logic that reads a schema of tables identified in metadata" and "a logic that automatically generates and caches prepare[d] SQL statements from a content metadata portion of document metadata according to the schema" should these terms be found to be means plus function.

165. A POSITA would understand that structure in the '575 patent corresponding to this term is the RTID software depicted in Figs. 12A-12E (as I described above in Section VI.B), and the description of this structure found throughout the patent, including in 29:62-36:47.

166. Additionally, a POSITA would understand that an algorithm for implementing "a logic that reads a schema of tables identified in metadata" and "a logic that automatically generates and caches prepare[d] SQL statements from a content metadata portion of document metadata according to the schema" is described at 16:30-50; 17:1-39; and 18:40-19:24.

g. Claim 12

167. Claim 12 recites (in relevant part):

12. The data services handler according to claim 1 further comprising:

...

a logic that invokes the real time information director asynchronously when an inbound data document is delivered via Java Message Service (JMS);

....

a logic that invokes the real time information director asynchronously when an outbound data document is requested via Java Message Service (JMS).

168. A POSITA would understand that this refers to the software on the data services handler, a specific structure for performing claimed functionality. This is consistent with the specification that refers consistently to the use of JMS and a Message Driven Bean to invoke the RTID asynchronously:

The real time information director is accessed synchronously via HyperText Transport Protocol (HTTP) and asynchronously via Java Message Service (JMS).

Id., 3:31-33.

The real time information director (RTID) 302 supplies a Message-Driven Bean (MDB) 334 to acquire inbound IDocs from JMS. Written in the Java Programming Language, a bean such as an enterprise bean is a server-side component that encapsulates business logic of an application. The business logic is code that performs the application. A message-driven bean is an enterprise bean enabling J2EE applications to process messages asynchronously.

Id., 10:1-8.

Referring to FIG. 3B, many applications receive documents from the RTSC data store 306 using a publish-subscribe paradigm. The subscription is configured in the real time information director 302. The subscription definition includes a triggering event, the document to be generated, and the name of a JMS queue or topic to which the document should be delivered. Applications directly receive the published document by reading the JMS queue or topic.

Id., 10:28-34.

Inbound documents can be delivered to the real time information director 302 synchronously, using HTTP, or asynchronously, using JMS. A

servlet can be supplied that invokes the real time information director 302 via an Enterprise Java Bean (EJB) whenever documents are delivered via HTTP. The real time information director 302 is invoked via a Message Driven Bean (MDB) whenever documents are delivered via JMS. In either case, the Bean is stateless, and serves only as a thin layer to access the services of the real time information director 302.

Id., 17:47-56.

169. A POSITA would have been familiar with the use of JMS and a Message Driven Bean to invoke other software applications, such as the RTID, sufficient to implement claim 12.

h. If Claim 12 were found to be means plus function, structure is nevertheless disclosed.

170. I have also been asked to consider what structure, if any, in the specification would correspond to claim 12's "a logic that invokes the real time information director asynchronously" should this term be found to be means plus function.

171. A POSITA would understand that structure in the '575 patent corresponding to this term is the JMS and a Message Driven Bean implemented as described above and per the description at Fig. 10B; 27:40-47 and 10:1-16.

H. "a business rules engine that..."

172. Claim 21 recites:

The data services handler according to claim 1 wherein the real time information director further comprises:

a business rules engine that defines a business object model capable of enabling access to data described by Document Metadata, performing computations on data relevant to business rules specified by the business object model, concluding a decision based on computation results, and executing the decision.

173. Similar to the "a logic that" terms above, it is clear here again that the "business rules engine" is software that runs as part of the real time information director. A POSITA would

understand, based on the claim language, that this is a specific software structure required by the claim. The *Microsoft Computer Dictionary* from 1999 notes, for example, that “the term engine is most often used in relation to a specific use” (Ex. 15 at -902):

engine *n.* A processor or portion of a program that determines how the program manages and manipulates data. The term *engine* is most often used in relation to a specific use; for example, a database engine contains the tools for manipulating a database, and a Web search engine has the ability to search World Wide Web indexes for matches to one or more key words entered by the user. *Compare* back-end processor, front-end processor.

174. The specification even discloses a specific “business rules engine,” namely Blaze

Advisor:

In a specific example, real time information director 802 can host a Blaze Advisor™ business rules engine 810. The real time information director 802 enables access to data that drives business rules 820. Typically a business object model is constructed to support Blaze. The business object model enables access to data, performs computations on the data that are relevant to the rules, and supports execution of decisions made by application of the rules. The real time information director 802 includes a pre-built business object model that is operational without customer modification. The business object model gives direct access to the data described by the document metadata 804 and also enables access to aggregates defined for a specified document type using Aggregate Wizards. The business object model may invoke the scoring engine using any scoring model relevant to the document.

With a capability to create custom aggregate functions, any calculation likely to be used by business rules can be performed without modifying the business object model, a capability that can tremendously shorten the time expended in building a solution using business rules 820.

Id., 26:28-50.

175. Blaze advisor was on sale in the prior art, and a POSITA would have known how to incorporate it into the structure of the RTID, especially when implemented the RTID was implemented in Java. As noted in Beth Bacheldor's *Personalized Shopping on the Internet*, Information Week (Oct. 4, 1999) (reproduced below), Blaze Advisor was available as "Java software tools" on sale for \$50,000:

Personalized Shopping On The Internet

Blaze and Zorder.com offer software that evaluates customers' use of Web sites

Companies such as YuSave.com, an upcoming online marketplace for consumers, want to personalize each customer's shopping experience. New products are becoming available to do it.

Blaze Software Inc. this week will introduce Blaze Advisor Rule Server and Blaze Advisor Innovator, Java software tools that let companies build personalization and self-service features into Web sites. The software can also link Web sites to other customer channels, such as call centers.

Last week, Zorder.com Inc. introduced its own software for personalizing the buying process, called Zorder Net Marketplace. "Our technology turns the Web site from being a passive catalog into a top salesperson," says Scott Geller, CEO of Zorder.com, formerly called BT

Squared Technologies Inc.

At the core of both companies' products are rules-based engines that automate Web-site activity by evaluating how customers use sites and what information they input. The software can instantly produce a response specific to a customer's query. The rules-based systems differ from personalization products that track click streams or the parts of sites that customers visit. "Rules can act as a brain for decision-making processes," says Lynne Harvey, a senior analyst with the Patricia Seybold Group.

Both companies say their software will help companies create proactive Web sites. For example, an online drugstore could recommend various over-the-counter drugs based on a customer's symptoms.

YuSave.com, which will begin selling thousands of consumer goods early next year, plans to use Zorder.com's software. "If you go to most of the Web sites out there, you can only shop the way they tell you to shop," says John Gunning, CEO of the Atlanta startup. "But there are a number of ways people shop. Some know exactly what they want; some people come to browse; some people shop by feature, value, and price. In the end, we want to deliver to them exactly the product they are looking for."

Zorder Net Marketplace is available now and is sold via per-processor or per-transaction licensing models. Pricing varies depending on configuration, but a per-processor licensing fee starts around \$100,000.

Blaze Advisor Rule Server is available, and Blaze Advisor Innovator will ship by year's end. The software starts at around \$50,000.

—Beth Bacheldor

Beth Bacheldor, *Personalized Shopping on the Internet*, Information Week (Oct. 4, 1999) (Ex. 17 at -622).

Blaze Software, Inc. (Nasdaq: BLZE) is a leading provider of rules- based e-business software enabling adaptable and personalized interaction that is consistent across all contact points. The Blaze Advisor Solutions Suite, written entirely in the Java programming language, is designed to give e-business applications more personalized interaction for users, which can lead to greater customer retention, decreased costs, and increased sales.

Business Wire, *Blaze Software Ships New Blaze Advisor Innovator, Letting Business Managers Adapt Systems to Changing Conditions without Programming* (June 6, 2000) (Ex. 16 at -623).

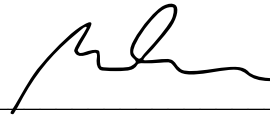
176. I note further that, were "a business rules engine" found to be a means plus function term, a POSITA would understand that the structure in the '575 patent corresponding to this term

is the business rules engine and associated business object model, as well as the examples of the operations of business rules, as described at Fig. 8; 18:25-30; 19:39-20:50; Fig. 7B; 25:52-56; 26:25-27:31 and 28:25-29:52.

VII. CONCLUSION

I reserve the right to supplement or amend my opinions in the future to respond to any arguments raised by Defendants or opinions offered by its expert(s) and to take into account new information that becomes available to me. I further reserve the right to respond to any questions raised by the Court in the event there is a hearing. At the hearing, I may use exhibits or demonstratives to aid my testimony. I have not prepared those exhibits or demonstrative aids at this time, other than the limited ones set forth in this declaration but expect that I will. I reserve the right to prepare and present those exhibits or demonstratives. I also reserve the right to provide a tutorial relating to the general topics contained within this declaration.

I declare under penalty of perjury of the laws of the United States that the foregoing is true and correct.



MARTIN FARACH-COLTON, M.D., PH.D.

Dated: December 20, 2024