IN THE UNITED STATES DISTRICT COURT

FOR THE DISTRICT OF DELAWARE

SUN MICROSYSTEMS, INC., Plaintiff, v. VERSATA ENTERPRISES, INC., VERSATA SOFTWARE, INC., VERSATA DEVELOPMENT GROUP, INC., VERSATA COMPUTER INDUSTRY SOLUTIONS, INC., VERSATA, INC., TRILOGY, INC., and NEXTANCE, INC., Defendants.	CIVIL ACTION NO. _____ **JURY TRIAL DEMANDED**

COMPLAINT

Plaintiff, Sun Microsystems, Inc., for its Complaint against Defendants, Versata Enterprises, Inc., Versata Software, Inc., Versata Development Group, Inc., Versata Computer Industry Solutions, Inc., Versata, Inc., Trilogy, Inc., and Nextance, Inc., alleges upon personal knowledge with respect to itself and its own acts, and upon information and belief with respect to all other matters, as follows:

THE PARTIES

1. Plaintiff, Sun Microsystems, Inc. ("Sun"), is a Delaware corporation with its principal place of business in Santa Clara, California.

2. Defendants, Versata Enterprises, Inc., Versata Software, Inc., Versata Development Group, Inc., Versata Computer Industry Solutions, Inc., Versata, Inc., Trilogy, Inc., and Nextance, Inc., are all Delaware corporations with their principal places of business in Austin, Texas. Defendants are commonly owned, have overlapping business relationships, and have acted in concert in connection with the infringement of Sun's patents alleged herein.

JURISDICTION AND VENUE

3. This Court has subject matter jurisdiction over this action pursuant to 28 U.S.C. §§ 1331 and 1338, in that this is a civil action for patent infringement arising under the Patent Laws of the United States, Title 35, United States Code. This Court has personal jurisdiction over Defendants by virtue of, among other things, their incorporation in Delaware.

4. Venue is proper in this district under 28 U.S.C. §§ 1391 and 1400.

COUNT ONE
(Patent Infringement – U.S. Patent No. 6,832,223)

5. Sun is the current assignee, and the sole and exclusive owner, of U.S. Patent No. 6,832,223 (the "'223 patent"), entitled "Method and System for Facilitating Access to a Lookup Service." The '223 patent was duly and legally issued by the United States Patent and Trademark Office ("PTO") on December 14, 2004. A copy of the '223 patent is attached hereto as Exhibit A.

6. In violation of 35 U.S.C. §§ 271(a), Defendants have made, used, offered for sale and/or sold in the United States a system that infringes one or more claims of the '223 patent.

7. Defendants also have actively induced the infringement of and/or contributed to the infringement of one or more claims of the '223 patent in violation of 35 U.S.C. §§ 271(b) & (c).

8. Defendants' infringement of the '223 patent has been willful and wanton.

9. Sun has suffered and will continue to suffer serious irreparable injury unless Defendants' infringement of the '223 patent is enjoined.

COUNT TWO
(Patent Infringement – U.S. Patent No. 5,870,719)

10. Sun is the current assignee, and the sole and exclusive owner, of U.S. Patent No. 5,870,719 (the "'719 patent"), entitled "Platform-Independent, Usage-Independent, and Access-Independent Distributed Quote Configuration System." The '719 patent was duly and legally issued by the PTO on February 9, 1999. A copy of the '719 patent is attached hereto as Exhibit B.

11. In violation of 35 U.S.C. § 271(a), Defendants have made, used, offered for sale and/or sold in the United States a system that infringes one or more claims of the '719 patent.

12. Defendants also have actively induced the infringement of, contributed to the infringement of one or more claims of the '719 patent in violation of 35 U.S.C. §§ 271(b) & (c).

13. Defendants' infringement of the '719 patent has been willful and wanton.

14. Sun has suffered and will continue to suffer serious irreparable injury unless Defendant's infringement of the '719 patent is enjoined.

COUNT THREE
(Patent Infringement – U.S. Patent No. 5,963,950)

15. Sun is the current assignee, and the sole and exclusive owner, of U.S. Patent No. 5,963,950 (the "'950 patent"), entitled "Method and System for Facilitating Access to Selectable Elements on a Graphical User Interface." The '950 patent was duly and legally issued by the PTO on October 5, 1999. A copy of the '950 patent is attached hereto as Exhibit C.

16. In violation of 35 U.S.C. § 271(a), Defendants have made, used, offered for sale and/or sold in the United States a system that infringes one or more claims of the '950 patent.

17. Defendants also have actively induced the infringement of, contributed to the infringement of one or more claims of the '950 patent in violation of 35 U.S.C. §§ 271(b) & (c).

18. Defendants' infringement of the '950 patent has been willful and wanton.

19. Sun has suffered and will continue to suffer serious irreparable injury unless Defendants' infringement of the '950 patent is enjoined.

COUNT FOUR
(Patent Infringement – U.S. Patent No. 5,761,662)

20. Sun is the current assignee, and the sole and exclusive owner, of U.S. Patent No. 5,761,662 (the "'662 patent"), entitled "Personalized Information Retrieval Using User-Defined Profile." The '662 patent was duly and legally issued by the PTO on June 2, 1998. A copy of the '662 patent is attached hereto as Exhibit D.

21. In violation of 35 U.S.C. § 271(a), Defendants have made, used, offered for sale and/or sold in the United States a system that infringes one or more claims of the '662 patent.

22. Defendants also have actively induced the infringement of, contributed to the infringement of one or more claims of the '662 patent in violation of 35 U.S.C. §§ 271(b) & (c).

23. Defendants' infringement of the '662 patent has been willful and wanton.

24. Sun has suffered and will continue to suffer serious irreparable injury unless Defendants' infringement of the '662 patent is enjoined.

WHEREFORE, Sun respectfully requests that this Court enter judgment in its favor and against Defendants and grant the following relief:

A. A judgment that Defendants have infringed the '223, '719, '950, and '662 patents in violation of 35 U.S.C. §§ 271 (a), (b) and (c);

B. A judgment that Defendants' infringement of the '223, '719, '950, and '662 patents has been willful and wanton;

C. A preliminary and permanent injunction, pursuant to 35 U.S.C. § 283, enjoining Defendants, and all persons in active concert or participation with Defendants, from any further acts of infringement, inducement of infringement, or contributory infringement of the '223, '719, '950, and '662 patents;

D. An order, pursuant to 35 U.S.C. § 284, awarding Sun damages adequate to compensate Sun for Defendants' infringement of the '223, '719, '950, and '662 patents, in an amount to be determined at trial, but in no event less than a reasonable royalty;

E. An order, pursuant to 35 U.S.C. § 284, and based on Defendants' willful and wanton infringement of the '223, '719, '950, and '662 patents, trebling all damages awarded to Sun;

F. An order, pursuant to 35 U.S.C. § 284, awarding to Sun interest on the damages and its costs incurred in this action;

G. An order, pursuant to 35 U.S.C. § 285, finding that this is an exceptional case and awarding to Sun its reasonable attorneys' fees incurred in this action; and,

H. Such other and further relief as this Court may deem just and proper.

DEMAND FOR JURY TRIAL

Sun demands trial by jury on all issues and causes of action properly tried by jury, pursuant to Federal Rule of Civil Procedure 38.

DATED: November 30, 2007

Respectfully submitted,



Paul J. Lockwood (Del. I.D. No. 3369)
Michael A. Barlow (Del. I.D. No. 3928)
SKADDEN, ARPS, SLATE,
MEAGHER & FLOM LLP
One Rodney Square
P.O. Box 636
Wilmington, Delaware 19899
Tel: (302) 651-3000
Fax: (302) 651-3001
E-mail:plockwoo@skadden.com
mbarlow@skadden.com

Attorneys for Plaintiff
Sun Microsystems, Inc.

OF COUNSEL:

Jeffery G. Randall
David W. Hansen
SKADDEN, ARPS, SLATE, MEAGHER
& FLOM LLP
525 University Avenue, Suite 1100
Palo Alto, California 94301
Tel: (650) 470-4500
Fax: (650) 470-4750

EXHIBIT A

US006832223B1

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

(10) **Patent No.: US 6,832,223 B1**
(45) **Date of Patent: Dec. 14, 2004**

(54) **METHOD AND SYSTEM FOR FACILITATING ACCESS TO A LOOKUP SERVICE**

(75) Inventors: **Robert W. Scheifler**, Somerville, MA (US); **Ann M. Wollrath**, Groton, MA (US); **James H. Waldo**, Dracut, MA (US)

(73) Assignee: **Sun Microsystems, Inc.**, Santa Clara, CA (US)

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

(21) Appl. No.: **09/044,826**

(22) Filed: **Mar. 20, 1998**

Related U.S. Application Data

(63) Continuation-in-part of application No. 08/636,706, filed on Apr. 23, 1996
(60) Provisional application No. 60/076,048, filed on Feb. 26, 1998.

(51) **Int. Cl.**[7] **G06F 17/00**
(52) **U.S. Cl.** **707/10**; 709/200; 707/200; 713/150
(58) **Field of Search** 709/200–232, 709/234; 707/1–206; 713/150–153

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,430,699	A	2/1984	Segarra et al.	364/200
4,491,946	A	1/1985	Kryskow, Jr. et al.	
4,558,413	A	12/1985	Schmidt et al.	364/300
4,567,359	A	1/1986	Lockwood	
4,713,806	A	12/1987	Oberlander et al.	
4,809,160	A	2/1989	Mahon et al.	
4,823,122	A	4/1989	Mann et al.	
4,939,638	A	7/1990	Stephenson et al.	
4,956,773	A	9/1990	Saito et al.	
4,992,940	A	2/1991	Dworkin	

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	0 300 516	A	1/1989
EP	0 384 339	A	8/1990
EP	0 472 874	A1	3/1992
EP	0 474 340	A	3/1992
EP	0 497 022	A1	8/1992
EP	0 555 997	A	8/1993
EP	0 565 849	A	10/1993

(List continued on next page.)

OTHER PUBLICATIONS

Moller et al., Enhancing Jini's lookup service using XML-–based service templates, Technology of Object–Oriented Languages and Systems, 2001, TOOLS 38, Proceedings, Mar. 12–14, 2001, pp. 19–31.*

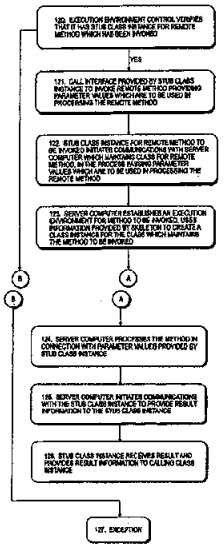
(List continued on next page.)

Primary Examiner—David Jung
(74) *Attorney, Agent, or Firm*—Finnegan, Henderson, Farabow, Garrett & Dunner, L.L.P.

(57) **ABSTRACT**

Methods and systems are provided that facilitate access to a service via a lookup service. A lookup service defines a network's directory of services and stores references to these services. A client desiring use of a service on the network accesses the lookup service, which returns the stub information that facilitates the user's access of the service. The client uses the stub information to access the service.

27 Claims, 9 Drawing Sheets



U.S. PATENT DOCUMENTS

5,088,036	A		2/1992	Ellis et al.	395/425
5,101,346	A		3/1992	Ohtsuki	
5,109,486	A		4/1992	Seymour	
5,187,787	A		2/1993	Skeen et al.	395/600
5,218,699	A		6/1993	Brandle et al.	
5,253,165	A		10/1993	Leiseca et al.	
5,257,369	A		10/1993	Skeen et al.	395/650
5,293,614	A		3/1994	Ferguson et al.	395/600
5,297,283	A		3/1994	Kelly et al.	
5,307,490	A		4/1994	Davidson et al.	709/203
5,311,591	A		5/1994	Fischer	
5,319,542	A		6/1994	King, Jr. et al.	
5,327,559	A		7/1994	Priven et al.	
5,339,435	A		8/1994	Lubkin et al.	
5,386,568	A		1/1995	Wold et al.	
5,390,328	A		2/1995	Frey et al.	
5,392,280	A		2/1995	Zheng	370/60
5,423,042	A		6/1995	Jalili et al.	
5,440,744	A		8/1995	Jacobson et al.	
5,448,740	A		9/1995	Kiri et al.	
5,452,459	A		9/1995	Drury et al.	395/700
5,455,952	A		10/1995	Gjovaag	
5,471,629	A		11/1995	Risch	
5,475,792	A		12/1995	Stanford et al.	
5,475,817	A		12/1995	Waldo et al.	
5,475,840	A		12/1995	Nelson et al.	
5,481,721	A		1/1996	Serlet et al.	
5,504,921	A		4/1996	Dev et al.	
5,511,196	A		4/1996	Shackelford et al.	
5,511,197	A	*	4/1996	Hill et al.	709/200
5,524,244	A		6/1996	Robinson et al.	
5,544,040	A		8/1996	Gerbaulet	
5,548,726	A		8/1996	Pettus	395/200.09
5,553,282	A		9/1996	Parrish et al.	
5,555,367	A		9/1996	Premerlani et al.	
5,555,427	A		9/1996	Aoe et al.	395/800
5,557,798	A		9/1996	Skeen et al.	395/650
5,560,003	A		9/1996	Nilsen et al.	395/600
5,561,785	A		10/1996	Blandy et al.	395/497.01
5,577,231	A		11/1996	Scalzi et al.	
5,592,375	A		1/1997	Salmon et al.	
5,594,921	A		1/1997	Pettus	395/831
5,603,031	A		2/1997	White et al.	395/683
5,617,537	A		4/1997	Yamada et al.	
5,628,005	A		5/1997	Hurvig	
5,640,564	A		6/1997	Hamilton et al.	
5,644,768	A		7/1997	Periwal et al.	395/672
5,652,888	A		7/1997	Burgess	
5,655,148	A		8/1997	Richman et al.	395/828
5,659,751	A		8/1997	Heninger	395/685
5,664,110	A		9/1997	Green et al.	
5,664,111	A		9/1997	Nahan et al.	
5,666,493	A		9/1997	Wojcik et al.	
5,671,225	A		9/1997	Hooper et al.	
5,671,279	A		9/1997	Elgamal	
5,675,796	A		10/1997	Hodges et al.	
5,675,797	A		10/1997	Chung et al.	
5,680,573	A		10/1997	Rubin et al.	
5,680,617	A		10/1997	Gough et al.	
5,684,955	A		11/1997	Meyer et al.	
5,689,709	A		11/1997	Corbett et al.	
5,694,551	A		12/1997	Doyle et al.	
5,706,435	A		1/1998	Barbará et al.	
5,706,502	A		1/1998	Foley et al.	395/610
5,710,887	A		1/1998	Chelliah et al.	
5,715,314	A		2/1998	Payne et al.	
5,721,832	A		2/1998	Westrope et al.	
5,724,540	A		3/1998	Kametani	
5,724,588	A	*	3/1998	Hill et al.	709/200
5,727,048	A		3/1998	Hiroshima et al.	
5,727,145	A	*	3/1998	Nessett et al.	709/200
5,729,594	A		3/1998	Klingman	
5,737,607	A		4/1998	Hamilton et al.	395/701
5,742,768	A		4/1998	Gennaro et al.	
5,745,678	A		4/1998	Herzberg et al.	395/186
5,745,695	A		4/1998	Gilchrist et al.	395/200.57
5,745,703	A		4/1998	Cejtin et al.	395/200.68
5,745,755	A		4/1998	Covey	395/619
5,748,897	A		5/1998	Katiyar	395/200.49
5,754,849	A		5/1998	Dyer et al.	395/612
5,757,925	A		5/1998	Faybishenko	380/49
5,758,328	A		5/1998	Giovannoli	
5,758,344	A		5/1998	Prasad et al.	707/10
5,761,656	A		6/1998	Ben-Shachar	707/4
5,764,897	A		6/1998	Khalidi	395/200.31
5,768,532	A		6/1998	Megerian	395/200.75
5,774,551	A		6/1998	Wu et al.	380/25
5,774,729	A		6/1998	Carney et al.	
5,778,179	A		7/1998	Kanai et al.	
5,778,187	A		7/1998	Monteiro et al.	395/200.61
5,778,228	A		7/1998	Wei	395/684
5,778,368	A		7/1998	Hogan et al.	707/10
5,784,560	A		7/1998	Kingdon et al.	395/200.31
5,787,425	A		7/1998	Bigus	707/6
5,787,431	A		7/1998	Shaughnessy	707/100
5,790,548	A		8/1998	Sistanizadeh et al.	370/401
5,790,677	A		8/1998	Fox et al.	
5,794,207	A		8/1998	Walker et al.	
5,802,367	A		9/1998	Held et al.	395/685
5,805,805	A		9/1998	Civanlar et al.	
5,808,911	A		9/1998	Tucker et al.	364/559
5,809,144	A		9/1998	Sirbu et al.	
5,809,507	A	*	9/1998	Cavanaugh, III	709/200
5,812,819	A		9/1998	Rodwin et al.	395/500
5,813,013	A		9/1998	Shakib et al.	707/102
5,815,149	A		9/1998	Mutschler, III et al.	345/335
5,815,709	A	*	9/1998	Waldo et al.	709/200
5,815,711	A		9/1998	Sakamoto et al.	395/701
5,818,448	A		10/1998	Katiyar	345/335
5,829,022	A		10/1998	Watanabe et al.	711/118
5,832,219	A		11/1998	Pettus	395/200.33
5,832,529	A		11/1998	Wollrath et al.	
5,832,593	A		11/1998	Wurst et al.	29/750
5,835,737	A		11/1998	Sand et al.	395/293
5,842,018	A		11/1998	Atkinson et al.	395/700
5,844,553	A		12/1998	Hao et al.	345/329
5,845,090	A		12/1998	Collins, III et al.	395/200.51
5,845,129	A		12/1998	Wendorf et al.	395/726
5,850,442	A		12/1998	Muftic	
5,860,004	A	*	1/1999	Fowlow et al.	709/200
5,860,153	A		1/1999	Matena et al.	711/216
5,864,862	A		1/1999	Kriens et al.	707/103
5,864,866	A		1/1999	Henckel et al.	707/103
5,872,928	A		2/1999	Lewis et al.	395/200.52
5,872,973	A		2/1999	Mitchell et al.	395/685
5,875,335	A		2/1999	Beard	395/705
5,878,411	A		3/1999	Burroughs et al.	707/4
5,884,024	A		3/1999	Lim et al.	395/187.01
5,884,079	A		3/1999	Furusawa	395/701
5,887,134	A		3/1999	Ebrahim	395/200.3
5,889,951	A		3/1999	Lombardi	395/200.49
5,890,158	A		3/1999	House et al.	707/10
5,892,904	A		4/1999	Atkinson et al.	395/187.01
5,913,029	A		6/1999	Shostak	395/200.33
5,933,497	A		8/1999	Beetcher et al.	380/4
5,933,647	A		8/1999	Aronberg et al.	395/712
5,935,249	A		8/1999	Stern et al.	713/201
5,940,827	A		8/1999	Hapner et al.	707/8
5,944,793	A		8/1999	Islam et al.	709/220
5,946,485	A		8/1999	Weeren et al.	395/703
5,946,694	A		8/1999	Copeland et al.	707/103

5,949,998	A	9/1999	Fowlow et al.	395/701
5,951,652	A	9/1999	Ingrassia, Jr. et al.	709/248
5,956,509	A	9/1999	Kevner	395/684
5,961,582	A	10/1999	Gaines	
5,963,924	A	10/1999	Williams et al.	
5,963,947	A	10/1999	Ford et al.	707/10
5,966,531	A	10/1999	Skeen et al.	395/683
5,969,967	A	10/1999	Aahlad et al.	364/131
5,974,201	A	10/1999	Chang et al.	382/305
5,978,484	A	11/1999	Apperson et al.	380/25
5,982,773	A	11/1999	Nishimura et al.	
5,987,506	A	11/1999	Carter et al.	709/213
5,999,179	A	12/1999	Kekic et al.	345/349
5,999,988	A	12/1999	Pelegri-Llopart et al.	709/304
6,003,050	A	12/1999	Silver et al.	
6,003,763	A	12/1999	Gallagher et al.	235/379
6,009,103	A	12/1999	Woundy	370/401
6,009,413	A	12/1999	Webber et al.	
6,016,496	A	1/2000	Roberson	707/103
6,016,516	A	1/2000	Horikiri	
6,023,586	A	2/2000	Gaisford et al.	395/712
6,026,414	A	2/2000	Anglin	707/204
6,031,977	A	2/2000	Pettus	395/200.6
6,032,151	A	2/2000	Arnold et al.	
6,044,381	A	3/2000	Boothby et al.	
6,052,761	A	4/2000	Hornung et al.	711/141
6,058,383	A	5/2000	Narasimhalu et al.	705/44
6,061,699	A	5/2000	DiCecco et al.	707/513
6,061,713	A	5/2000	Bharadhwaj	709/203
6,067,575	A	5/2000	McManis et al.	709/300
6,085,255	A	7/2000	Vincent et al.	709/238
6,093,216	A	7/2000	Adl-Tabatabai et al.	
6,108,346	A	8/2000	Doucette et al.	370/450
6,134,603	A	10/2000	Jones et al.	
6,182,083	B1	1/2001	Scheifler et al.	
6,185,602	B1	2/2001	Bayrakeri	709/204
6,185,611	B1	2/2001	Waldo et al.	709/221
6,199,116	B1	3/2001	May et al.	
6,216,138	B1	4/2001	Wells et al.	707/502
6,216,158	B1	4/2001	Luo et al.	
6,226,746	B1	5/2001	Scheifler	713/200
6,243,716	B1	6/2001	Waldo et al.	
6,247,091	B1	6/2001	Lovett	710/260
6,253,256	B1	6/2001	Wollrath et al.	
6,263,350	B1	7/2001	Wollrath et al.	
6,263,379	B1	7/2001	Atkinson et al.	
6,272,559	B1	8/2001	Jones et al.	
6,282,295	B1	8/2001	Young et al.	380/286
6,282,568	B1	8/2001	Sondur et al.	709/223
6,339,783	B1	1/2002	Horikiri	
6,343,308	B1	1/2002	Marchesseault	
6,385,643	B1	5/2002	Jacobs et al.	
6,687,717	B1 *	2/2004	Hamilton et al.	707/203
6,697,865	B1 *	2/2004	Howard et al.	707/9
6,701,438	B1 *	3/2004	Prabandham et al.	707/10

FOREIGN PATENT DOCUMENTS

EP	0 569 195	A2	11/1993
EP	0 625 750	A	11/1994
EP	0 635 792	A	1/1995
EP	0 651 328	A	5/1995
EP	0 660 231	A	6/1995
EP	0 697 655	A2	2/1996
EP	0 718 761	A	6/1996
EP	0 767 432	A2	4/1997
EP	0 778 520	A2	6/1997
EP	0 794 493	A2	9/1997
EP	0 803 810	A2	10/1997
EP	0 803 811	A2	10/1997
EP	0 805 393	A2	11/1997
EP	0 810 524	A	12/1997
EP	0 817 020	A	1/1998
EP	0 817 022	A2	1/1998
EP	0 817 025	A2	1/1998
EP	0 836 140	A2	4/1998
EP	0 351 536	A3	1/1999
GB	2 253 079	A	8/1992
GB	2 262 825	A	6/1993
GB	2 305 087	A	3/1997
JP	11-45187		2/1999
WO	WO92/07335		4/1992
WO	WO92/09948		6/1992
WO	WO93/25962	A1	12/1993
WO	WO94/03855		2/1994
WO	WO96/03692	A	2/1996
WO	WO96/10787		4/1996
WO	WO96/18947		6/1996
WO	WO96/24099		8/1996
WO	WO98/02814		1/1998
WO	WO98/04971		2/1998

OTHER PUBLICATIONS

Morgan, Jiini to the rescue, Spectrum IEEE, vol. 37, Issue 4, Apr. 2000, pp. 44–49.*

Shin et al., Grapes: topology–based hierarchical virtual network for peer–to–peer lookup services, Parallel Processing Workshops, 2002, Proceedings, International Conference on, Aug. 18–21, 2002, pp. 159–164.*

Jacob, "The use of distributed objects and dynamic interfaces in a wide–are transaction environment", SIGCOMM '95 Workshop on Middleware: Cambridge Massachusetts, Aug. 1995.*

Alexander, et al., "Active Bridging", Proceedings of the ACM/SIGCOMM'97 Conference, Cannes, France, 9/97.

Anonymous: "Change–Notification Service for Shared Files", IBM Technical Disclosure Bulletin, vol. 36, No. 8, pp. 77–82, Aug. 1993, XP002108713, New York, US.

Anonymous: "Resource Preemption for Priority Scheduling." Nov. 1973. IBM Technical Disclosure Bulletin, vol. 16, No. 6, p. 1931 XP002109435 New York, US.

Beech et al., "Object Databases as Generalizations of Relational Databases," Computer Standards & Interfaces, vol. 13, Nos. 1/3, pp. 221–230, (Jan. 1991) Amsterdam, NL.

Bertino et al., "Object–Oriented Database Management Systems: Concepts and Issues," Computer, vol. 24, No. 4, pp. 33–47, (Apr. 1991), Los Alamitos, CA.

Betz, Mark; "Interoperable objects: laying the foundation for distributed object computing"; Dr. Dobb's. Journal, vol. 19, No. 11, p. 18(13); (Oct. 1994).

Bevan, D.I., "An Efficient Reference Counting Solution To The Distributed Garbage Collection Problem", Parall Computing, NL, Elsevier Publishers, Amsterdam, vol. 9, No. 2, pp. 179–192, Jan. 1989.

Birrell et al., "Implementing Remote Procedure Calls", ACM Transactions on Computer Systems, vol. 2, No. 1, Feb. 1984, pp. 39–59.

Deux O et al: "The 02 System" Communications of the Association for Computing Machinery, vol. 34, No. 10, Oct. 1, 1991, pp. 34–48.

Drexler, K. Eric, et al., "Incentive Engineering for Computational Resource Management," The Ecology of Computation, Elsevier Science Publishers B.V., 1988, pp. 231–266.

Droms, R., "RFC 1541 Dynamic Host Configuration Protocol", HTTP://WWW.CIS.OHIO–STATE.EDU/HTBIN/RFC/RFC1541.HTML, Oct. 1993, pp. 1–33.

Emms J: "A Definition of an Access Control Systems Language" Computer Standards and Interfaces, vol. 6, No. 4, Jan. 1, 1987, pp. 443–454.
Gosling et al., "The Java (TM) Language Specification", Addison–Wesley, 1996.
Gottlob et al., "Extending Object–Oriented Systems with Roles," ACM Transactions on information systems, vol. 14, No. 3, pp. 268–296 (Jul. 1996).
Guth, Rob: "JavaOne: Sun to Expand Java Distributed Computing Effort", "HTTP://WWW.SUNWORLD.COM/SWOL–02–1998/SWOL–02–SUNSPOTS.HTML,"
XP–002109935, P.1, Feb. 20, 1998.
Hamilton et al., "Subcontract: a flexible base for distributed programming"; Proceedings of 14th Symposium of Operating System Principles; (Dec. 1993).
Hartman, J., Manber, U., et al., Liquid Software: A new paradigm for networked systems, Technical Report 96–11, Department of Comp. Sci., Univ. of Arizona, 6/96.
Hunt, N., "IDF: A Graphical Data Flow Programming Language for Image Processing and Computer Vision", Proceedings of the International Conference on Systems, Man, and Cybernetics, Los Angeles, Nov. 4–7, pp. 351–360, (1990).
IBM (TM) Technical Disclosure Bulletin, "Object Location Algorithm," vol. 36, No. 09B, pp. 257–258, Sep. 1993.
IBM (TM) Technical Disclosure Bulletin, "Retrieval of Qualified Variables Using Extendible Hashing," vol. 36, No. 12, pp. 301–303, Dec. 1993.
IBM: "Chapter 6—Distributed SOM (DSOM)" Somobjects Developer Toolkit Users Guide, Version 2.1, Oct. 1994 (1994–10), pp. 6–1–6–90.
Jones, Richard, et al., "Garbage Collection: Algorithms for Automatic Dynamic Memory Management," pp. 165–175, John Wiley & Sons, 1996.
Kay, Michael H. et al., "An Overview of the Raleigh Object–Oriented Database System", ICL Technical Journal, vol. 7, No. 4, pp. 780–798, (Nov. 1991), Oxford, GB.
Kougiouris et al.; "Support for Space Efficient Object Invocation in Spring"; (Sep. 1994).
Lindholm et al., "The Java (TM) Virtual Machine Specification", Addison Wesley, 1996.
Mitchell et al.; "An Overview of the Spring System"; (Feb. 1994).
Orfali R. et al., "The Essential Distributed Objects Survival Guide," Chapter 11: Corba Commercial ORBs, pp. 203–215, John Wiley & Sons, Inc., (1996).
Riggs Roger et al., "Pickling State in the Java (TM) System," USENIX Association Conference on Object–Oriented Technologies and Systems, XP–002112719, Jun. 17–21, 1996, pp. 241–250.
Rosenberry et al., "Understanding DCE"; Chapters 1–3, 6; (1992).
Waldo J et al: "Events in an RPC based distributed system" Proceedings of the 1995 Usenix Technical Conference, Proceedings Usenix Winter 1995 Technical Conference, New Orleans, LA. USA, Jan. 16–20, 1995, pp. 131–142.
Wilson, P.R., et al., "Design of the Opportunistic Garbage Collector," Proceedings of the Object Oriented Programming Systems Languages And Applications Conference, New Orleans, vol. 24, No. 10, Oct. 1989.
Wu, Xuequn, "A Type system for an Object–Oriented Database System," Proceedings of the International Computer Software and Applications Conference (COMPSAC), pp. 333–338, Sep. 11–13, 1991, Tokyo, Japan.
Yemini, Y. and S. da silva, "Towards Programmable Networks", IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, L'Aquila, Italy, 10/96.
Aldrich et al., "Providing Easier Access to Remote Objects in Client–Server Systems," System Sciences, 1998, Proceedings of the 31st Hawaii Internat'l. Conference, Jan. 6–9, 1998, pp. 366–375.
Aldrich et al., "Providing Easier Access to Remote Objects in Distributed Systems," Calif. Institute of Technology, www.cs.caltech.edu/%7Ejedi/paper/jedipaper.html, Nov. 21, 1997.
Burns et al., "An Analytical Study of Opportunistic Lease Renewal," Distributed Computing Systems, 21st International Conference, pp. 146–153, Apr. 2000.
Dollimore et al., "The Design of a System for Distributing Shared Objects," The Computer Journal, No. 6, Cambridge, GB, Dec. 1991.
Fleisch et al., "High Performance Distributed Objects Using Distributed Shared Memory & Remote Method Invocation," System Sciences, 1998, Proceedings of the 31st Hawaii Internat'l. Conference, Jan. 6–9, 1998, pp. 574–578.
Gray et al., "Leases: An Efficient Fault–Tolerant Mechanism for Distributed File Cache Consistency," Proceedings of the 12th ACM Symposium on Operating Systems Principles, pp. 202–210, 1989.
Guyennet et al., "A New Consistency Protocol Implemented in the CAliF System," IEEE, 1094–7256/97, pp. 82–87, 1997.
Guyennet et al., "Distributetd Shared Memory Layer for Cooperative Work Applications," IEEE, 0742–1303/97, pp. 72–78, 1997.
Hoshi et al., "Allocation of the Cross–Connect Function in Leased Circuit Networks," 1992, ICC'92, conference record, SUPERCOMM/ICC '02, D a New World of Communications, IEEE International Conference, pp. 1408–1412.
IBM Technical Disclosure Bulletin, "Local Network Monitoring to Populate Access Agent Directory," vol. 36, No. 09A, pp. 403–405, Sep. 1993.
McGrath, "Discovery and Its Discontents: Discovery Protocols for Ubiquitous Computing," Presented at Center for Excellence in Space Data and Information Science, NASA Goddard Space Flight Center, Apr. 5, 2000.
MUX–Elektronik, Java 1.1 Interactive Course, www.lls.se/~mux/javaic.html, 1995.
Stevenson, "Token–Based Consistency of Replicated Servers," IEEE, CH2686–4/89/0000/0179, pp. 179–183, 1989.
Venners, "Jini Technology, Out of the Box," JAVAWORLD, 'Online!, pp. 1–4, Dec. 1998.
Yin et al., "Using Leases to Support Server Driven Consistency in Large–Scale Systems," Computer Services Department, University of Texas at Austin, p. 285–294, May 26–28, 1998.
Yin et al., "Volume Leases for Consistency in Large–Scale Systems," IEEE Transactions on Knowledge & Data Engineering, vol. 11, No. 4, pp. 563–576, Jul./Aug. 1999.
Amitabh Dave et al., "Proxies, Application Interfaces, and Distributed Systems," Proceedings International Workshop on Object Orientation in Operating Systems, pp. 212–220 (Sep. 24, 1992) XP002009478.
Howard et al., *Scale and Performance in a Distributed File System*, ACM Transactions on Computer Systems, vol. 6, No. 1, Feb. 1988, pp. 51–81.

Cardelli, *Obliq, A lightweight language for network objects*, Nov. 5, 1993, pp. 1–37.
Dijkstra, *Self–stabilizing Systems in Spite of Distributed Control*, Communications of the ACM, vol. 17, No. 11, Nov. 1974, pp. 643–644.
Ousterhout et al., *The Sprite Network Operating System*, Computer, IEEE, Feb. 1988, pp. 23–36.
Dourish, *A Divergence–Based Model of Synchrony and Distribution in Collaborative Systems*, Xerox Technical Report EPC–1194–102, 1994, pp. 1–10.
Sharrott et al., *ObjectMap: Integrating High Performance Resources into a Distributed Object–oriented Environment*, ICODP, 1995.
Birrell et al., *Grapevine: An Exercise in Distributed Computing*, Communication of the ACM, vol. 25, No. 4, Apr. 1982, pp. 260–274.
Transparent Network Computing, Locus Computing Corporation, Jan. 5, 1995.
Gray et al., *Leases: An Efficient Fault–Tolerant Mechanism for Distributed File Cache Consistency*, ACM, 1989, pp. 202–210.
Lamport et al., *The Byzantine Generals Problem*, ACM Transactions on Programming Languages and Systems, vol. 4, No. 3, Jul. 1982, pp. 382–401.
Dolev et al., *On the Minimal Synchronism Needed for Distributed Consensus*, Journal of the ACM, vol. 34, No. 1, Jan. 1987, pp. 77–97.
Mummert et al., *Long Term Distributed File Reference Tracing: Implementation and Experience*, Carnegie Mellon University School of Computer Science, Nov. 1994, pp. 1–28.
Gelernter et al., *Parallel Programming in Linda*, Yale University, Jan. 1985, pp. 1–21.
Cannon et al., *Adding Fault–Tolerant Transaction Processing to LINDA*, Software–Practice and Experience, vol. 24(5), May 1994, pp. 449–466.
Kambhatla et al., *Recovery with Limited Replay: Fault–Tolerant Processes in Linda*, Oregon Graduate Institute, Technical Report CSIE 90–019, Sep. 1990, pp. 1–16.
Anderson et al., *Persistent Linda: Linda + Transactions + Query Processing*, Proceedings of the 13th Symposium on Fault Tolerant Systems, 1994, pp. 93–109.
Gelernter, *Generative Communication in Linda*, ACM Transactions on Programming Languages and Systems, vol. 7, No. 1, Jan. 1985, pp. 80–112.
Carriero et al., *Distributed Data Structures in Linda*, Principals of Programming Language, 1986, pp. 1–16.
Pinakis, *Using Linda as the Basis of an Operating System Microkernel*, University of Western Australia, Department of Computer Science, Aug. 1993, pp. 1–165.
LINDA Database Search, Jul. 20, 1995, pp. 1–68.
Carriero et al., *Distributed Data Structures in Linda*, Yale Research Report YALEU/DCS/RR–438, Nov. 1985.
Agha et al., *Actorspaces: An Open Distributed Programming Paradigm*, University of Illinois, Report No. UIUCDCS–R–92–1766, Open Systems Laboratory TR No. 8, Nov. 1992, pp.1–12.
Ahmed et al., *A Program Building Tool for Parallel Applications*, Yale University, Dec. 1, 1993, pp. 1–23.
Liskov et al., *Distributed Object Management in Thor*, International Workshop on Distributed Object Management, 1992, pp. 12.
Birrell et al., *Network Objects*, DEC SRC Research Report 115, Feb. 28, 1994.
Birrell et al., *Distributed Gargage Collection for Network Objects*, DEC SRC Research Report 116, Dec. 15, 1993.
Wollrath et al., *A Distributed Object Model for the Java™ System*, USENIX Association, Conference on Object–Oriented Technologies and Systems, Jun. 17–21, 1996.
Harris et al., *Proposal for a General Java Proxy Class for Distributed Systems and Other Uses*, Netscape Communications Corp., Jun. 25, 1997.
Hamilton, *Java and the Shift to Net–Centric Computing* Computer, Aug. 1996, pp. 31–39.
Chung et al., *A 'Tiny' Pascal Compiler: Part 1: The P–Code Interpreter*, BYTE Publications, Inc., Sep. 1978.
Chung et al., *A 'Tiny' Pascal Compiler: Part 2: The P–Compiler*, BYTE Publications, Inc., Oct. 1978.
Thompson, *Regular Expression Search Algorithm*, Communications of the ACM, vol. II, No. 6, p. 149 et seq., Jun. 1968.
Mitchell et al., *Mesa Language Manual*, Xerox Corporation.
McDaniel, *An Analysis of a Mesa Instruction Set*, Xerox Corporation, May 1982.
Pier, *A Retrospective on the Dorado, A High–Performance Personal Computer*, Xerox Corporation, Aug. 1983.
Pier, *A Retrospective on the Dorado, A High–Performance Personal Computer*, IEEE Conference Proceedings, The 10th Annual international Symposium on Computer Architecture, 1983.
Krasner, *The Smalltalk–80 Virtual Machine*, BYTE Publications Inc., Aug. 1991, pp. 300–320.
Operating Systems Review, ACM Press, vol. 27, No. 5, Dec. 1993, pp. 217–230.
Remote Method Invocation Specification, Sun Microsystems, Inc., (1997), http://java.sun.com/products/jdk/1.1/docs/guide/rmi/spec/rmiTOC.doc.html.
Kolodner et al., "Atomic Garbage Collection: Managing a Stable Heap," ACM, 1989, pp. 15–25.
Osborn, "The Role of Polymorphism in Schema Evolution in an Object–Oriented Database," IEEE Transactions on Knowledge and Data Engineering, vol. 1, No. 3, Sep. 1989, pp. 310–317.
"Consumers Can View, Share Picture On–Line as Kodak Picture Network Goes 'Live'," Business Wire, Aug. 25, 1997, pp. 18–19.
"Eden Project Proposal," Department of Computer Science, University of Washington, Oct. 1980, Technical Report #80–10–01, cover and Foreword.
"ISN Dataweb Sells Software, Hardware," Datamation, Apr. 1, 1996, p. 40.
"Kodak DC220 And DC260 Digital Cameras Are Shipping To Retailers Across The Country Cameras Are Optimized for USB Interface Supported in Windows 98," Business Wire, Jun. 24, 1998, pp. 42–44.
"Kodak demonsrates leadership across entire photographic category," M2 Presswire, Feb. 13, 1998, pp. 31–35.
"Kodak PhotoNet Online Makes It a Snap to Share Summer Photos," PR Newswire, Jul. 2, 1998, pp. 63–64.
"Kodak Picture Network Sends Prints Home From the Holidays," Business Wire, Dec. 29, 1997, pp. 58–60.
"Photo processing made easy on the Internet; Storm Software and PictureVision Team up with Konica Corp.," Business Wire, Feb. 22, 1996, pp. 3–4.
"Seeing your photos a whole new way," Business Wire, Dec. 12, 1996, pp. 9–10.
"Webwatch: MCI Announces Internet Access,", Boardwatch Magazine, Jan. 1995.

Administrator's Guide, Netscape Enterprise Server, Version 3.0, Netscape Communications Corp., 1998.

Almes et al., "Edmas: A Locally Distributed Mail System," Department of Computer Science, University of Washington, Technical Report 83–07–01, Jul. 7, 1983, Abstract & pp. 1–17.

Almes et al., "Research in Integrated Distributed Computing," Department of Computer Science, University of Washington, Oct. 1979, pp. 1–42.

Almes et al., "The Eden System: A Technical Review," Department of Computer Science, University of Washington, Technical Report 83–10–05, Oct. 1983, pp. 1–25.

Almes, "Integration and Distribution in the Eden System," Department of Computer Science, University of Washington, Technical Report 83–01–02, Jan. 19, 1983, pp. 1–18 & Abstract.

Almes, "The Evolution of the Eden Invocation Mechanism," Department of Computer Science, University of Washington, Technical Report 83–01–03, Jan. 19, 1983, pp. 1–14 & Abstract.

Arnold, Ken, "The Jini Architecture: Dynamic Services in a Flexible Network," Sun Microsystems, Inc., Proceedings of the 36th ACM IEEE Design Automation Conference, Jun. 1999, pp. 157–162.

Bandrowski, "Stores Without Doors: Kiosks Generate New Profits," Corporate Computing, Oct. 1992, pp. 193–195.

Begole et al., "Transparent Sharing of Java Applets: A Replicated Approach," Oct. 1997, pp. 55–65.

Black et al., "A Language for Distributed Programming," Department of Computer Science, University of Washington, Technical Report 86–02–03, Feb. 1986, p. 10.

Black et al., "Distribution and Abstract Types in Emerald," University of Washington, Technical Report No. 85–08–05, Aug. 1985, pp. 1–10.

Black et al., "Object Structure in the Emerald System," University of Washington, Technical Report 86–04–03, Apr. 1986, pp. 1–14.

Black et al., "The Eden Project: A Final Report," Department of Computer Science, University of Washington, Technical Report 86–11–01, Nov. 1986, pp. 1–28.

Black, "Supporting Distributed Applications: Experience with Eden," Department of Computer Science, University of Washington, Technical Report 85–03–02, Mar. 1985, pp. 1–21.

Black, "The Eden Programming Language," Department of Computer Science, FR–35, University of Washington, Technical Report 85–09–01, Sep. 1985 (Revised Dec. 1985), pp. 1–19.

Black, "The Eden Project: Overview and Experiences," Department of Computer Science, University of Washington, EUUG, Autumn '86 Conference Proceedings, Manchester, UK, Sep. 22–25, 1986, pp. 177–189.

Braine et al., "Object–Flow," 1997, pp. 418–419.

Bruno, "Working the Web," Data Communications, Apr. 1997, pp. 50–60.

Ciancarini et al., "Coordinating Distributed Applets with Shade/Java," Feb. 1998, pp. 130–138.

Cohen, "Electronic Commerce," USC/Information Sciences Institute, Oct. 1989.

Conhaim, "Online shopping: a beginner's guide; includes related listing of videotex services," Link–Up, vol. 5, No. 6, p. 32, Nov. 1988.

Delcambre et al., "Simulation of the Object Flow Model: A Conceptual Modeling Language for Object–Driven Applications," 1993, pp. 216–225.

Design Project #2, Electronic Shopping at MIT, MIT Class 6.033 Handout 23, Massachusetts Institute of Technology, http://web.mit.edu/6.033/1995/handouts/html/h23.html, Spring 1995, pp. 1–6.

Ellsworth, "Boom Town," Internet World, Jun. 1995, pp. 33–35.

Estrin, "Inter–Organization Networks: Implications of Access Control Requirements for Interconnection Protocols," ACM, 1986, pp. 254–263.

Fleischer, "SkyMall's 'Supplier Network' Takes Flight", Retailtech, The Technology Magazine for Retail Executives, Apr. 1997.

Foley, "Managing Campus–Wide Information Systems: Issues and Problems," Capitalizing on Communication, ACM SIGUCCS XVII, 1989, pp. 169–174.

Fryxell, "eaasySABRE," Link–Up, May/Jun. 1996, pp. 10–11.

Gardner, "Kodak Follows Startup Into Online Photo Processing Business," Internet World, Sep. 8, 1997, pp. 5–6.

Gogan et al., "Open Market, Inc.: Managing in a Turbulent Environment," Harvard Business School Publishing, Aug. 29, 1996, pp. 1–30.

Goldberg et al., "Smalltalk–80—The Language and its Implementation," Xerox Palo Alto Research Center, 1983 (reprinted with corrections, Jul. 1985), pp. 1–720.

Hodges, Douglas, "Managing Object Lifetimes in OLE," Aug. 25, 1994, pp. 1–41.

Holman et al., "The Eden Shared Calendar System," Department of Computer Science, FR–35, University of Washington, Technical Report 85–05–02, Jun. 22, 1985, pp. 1–14.

Hsu, "Reimplementing Remote Procedure Calls," University of Washington, Thesis, Mar. 22, 1985, pp. 1–106.

Hutchinson, "Emerald: An Object–Based Language for Distributed Programming," a Dissertation, University of Washington, 1987, pp. 1–107.

Israel et al., "Authentication in Office System Internetworks," ACM Transactions on Office Information Systems, vol. 1, No. 3, Jul. 1983, pp. 193–210.

Jul et al., "Fine–Grained Mobility in the Emerald System," University of Washington, ACM Transactions on Computer Systems, vol. 6, No. 1, Feb. 1988, pp. 109–133.

Jul, "Object Mobility in a Distributed Object–Oriented System," a Dissertation, University of Washington, 1989, pp. 1–154 (1 page Vita).

Keller, "Smart Catalogs and Virtual Catalogs," Proceedings of the First USENIX Workshop of Electronic Commerce, USENIX Association, Jul. 11–12, 1995, pp. 125–131.

Kline et al., "TRADE'ex: The Stock Exchange of the Computer Industry," Harvard Business School Publishing, 1996, pp. 1–14.

Kodak Photonet Faq, PhotoNet Online, Jun. 14, 2002, pp. 1–3.

Koshizuka et al., "Window Real–Objects: A Distributed Shared Memory for Distributed Implementation of GUI Applications," Nov. 1993, pp. 237–247.

Kramer, "NETWATCH; The AJC's Daily Online Guide; Get the picture: Kodak will send photos to Web," The Atlanta Journal and Constitution, Sec. Features, p. 08C, Jun. 5, 1997.

Krasner et al., "Smalltalk–80: Bits of History, Words of Advice," 1983, Xerox Corporation, pp. 1–344.

Lampson et al., "Authentication in Distributed Systems: Theory and Practice," ACM Transactions n Computer Systems, vol. 10, No. 4, Nov. 1992, pp. 265–310.
Lansky, "Without APS, Photo Life Goes on Via Internet," Photographic Trade News, Aug. 1996, pp. 19–23.
Lavana et al., "Executable Workflows: A Paradigm for Collaborative Design on the Internet," Jun. 1997, 6 pages.
Lewis, "Pacific Bell, MCI to Expand Internet Service," The New York Times, sec. D, col. 1 at 3, Mar. 28, 1995.
LightSurf Instant Imaging—Press Releases, "Kodak And LightSurf Collaborate On Kodak Picture Center Online," LifeSurf Technologies Inc., Jun. 14, 2002, pp. 1–3.
Louwerse et al., "Data Protection Aspects in an Integrated Hospital Information System," North–Holland Computers & Security 3, 1984, pp. 286–294.
McEnaney, "Point–and–Click Memory Sharing; Launches PhotoNet online digital photography and imaging services," Photographic Trade News, Sec. p. 23, Jan. 1997.
Miller, "Web posting as a photo processing option," USA Today, Section: Life, p. 17D, Dec. 13, 1996.
Morris et al., "Andrew: A Distributed Personal Computing Environment," Communications of the ACM, vol. 29, No. 3, Mar. 1986, pp. 184–201.
O'Mahony, "Security Considerations in a Network Management Environment," IEEE Network, May/Jun. 1994, pp. 12–17.
Oppen et al., "The Clearinghouse: A Decentralized Agent for Locating Names Objects in a Distributed Environment," ACM Transactions on Office Information Systems, vol. 1, No. 3, Jul. 1983, pp. 230–253.
Petersen, "New But Not Improved," Direct Magazine, Nov. 1995.
Press Release, "Sun Goes Live With The Kodak Picture Network," Sun Microsystems, Inc., Jun. 14, 2002, pp. 1–2.
Proceedings of the Eighth Symposium on Operating Systems Principles, Dec. 14–16, 1981, ACM, Special Interest Group on Operating Systems, Association for Computing Machinery, vol. 15, No. 5, Dec. 1981, ACM Order No. 534810.
Proudfoot, "Replects: Data Replication in the Eden System," Department of Computer Science, University of Washington, Technical Report No. TR–85–12–04, Dec. 1985, pp. 1–156.
Pu, "Replication and Nested Transaction in the Eden Distributed System," Doctoral Dissertation, University of Washington, Aug. 6, 1986, pp. 1–179 (1 page Vita).
Raeder, "Is there a Prodigy in your future?," Database Searcher, vol. 5, No. 6, p. 18.
Ramm et al., "Exu—A System for Secure Delegation of Authority on an Insecure Network," Ninth System Administration Conference, 1995 LISA IX, Sep. 17–22, 1995, pp. 89–93.
Satyanarayanan, "Integrating Security in a Large Distributed System," ACM Transactions on Computer Systems, vol. 7, No. 3, Aug. 1989, pp. 247–280.
Schroeder et al., "Experience with Grapevine: The Growth of a Distributed System," ACM Transactions on Computer Systems, vol. 2, No. 1, Feb. 1984, pp. 3–23.
Senn, "Capitalizing on Electronic Commerce: The Role of the Internet in Electronic Markets," Information Systems Management, Summer 1996, pp. 15–24.
Steinke, "Design Aspects of Access Control in a Knowledge Base System," Computers & Security, 10, 1991, pp. 612–625.
Stern, "Industry Net," Link–Up, Mar./Apr. 1995, p. 10.
Tanenbaum et al., "Distributed Operating Systems," Computing Surveys, vol. 17, No. 4, Dec. 1985, pp. 419–470.
The Wall Street Journal, "Barclays Is Opening An 'Electronic Mall' For Internet Shopping," Tech. & Health Section at B2, Jun. 1, 1995.
The Wall Street Journal, "Prodigy Plans to Announce Internet 'Electronic Mail'," Tech. Section at B5, Nov. 27, 1995.
Trehan et al., "Toolkit for Shared Hypermedia on a Distributed Object Oriented Architecture," 1993, pp. 1–8.
Trommer, "Thomas Unveils Online Purchasing Network—Eases Product Sourcing And Ordering Through EDI," Electronic Buyers' News at 60, Dec. 11, 1995.
Van Den Berg et al., "Advanced Topics of a Computer Center Audit," North–Holland Computers & Security 3, 1984, pp. 171–185.
Van Der Lans, "Data Security in a Relational Database Environment," North–Holland Computers & Security 5, 1986, pp. 128–134.
Welz, "New Deals: A ripening Internet market, secure systems and digital currency are reshaping global commerce," Internet World, Jun. 1995, pp. 36–41.
Wobber et al., "Authentication in the Taos Operating System," ACM, 1993, pp. 256–269.
Wyatt, "Netscape Enterprise Server," Prima Publishing, 1996.

* cited by examiner

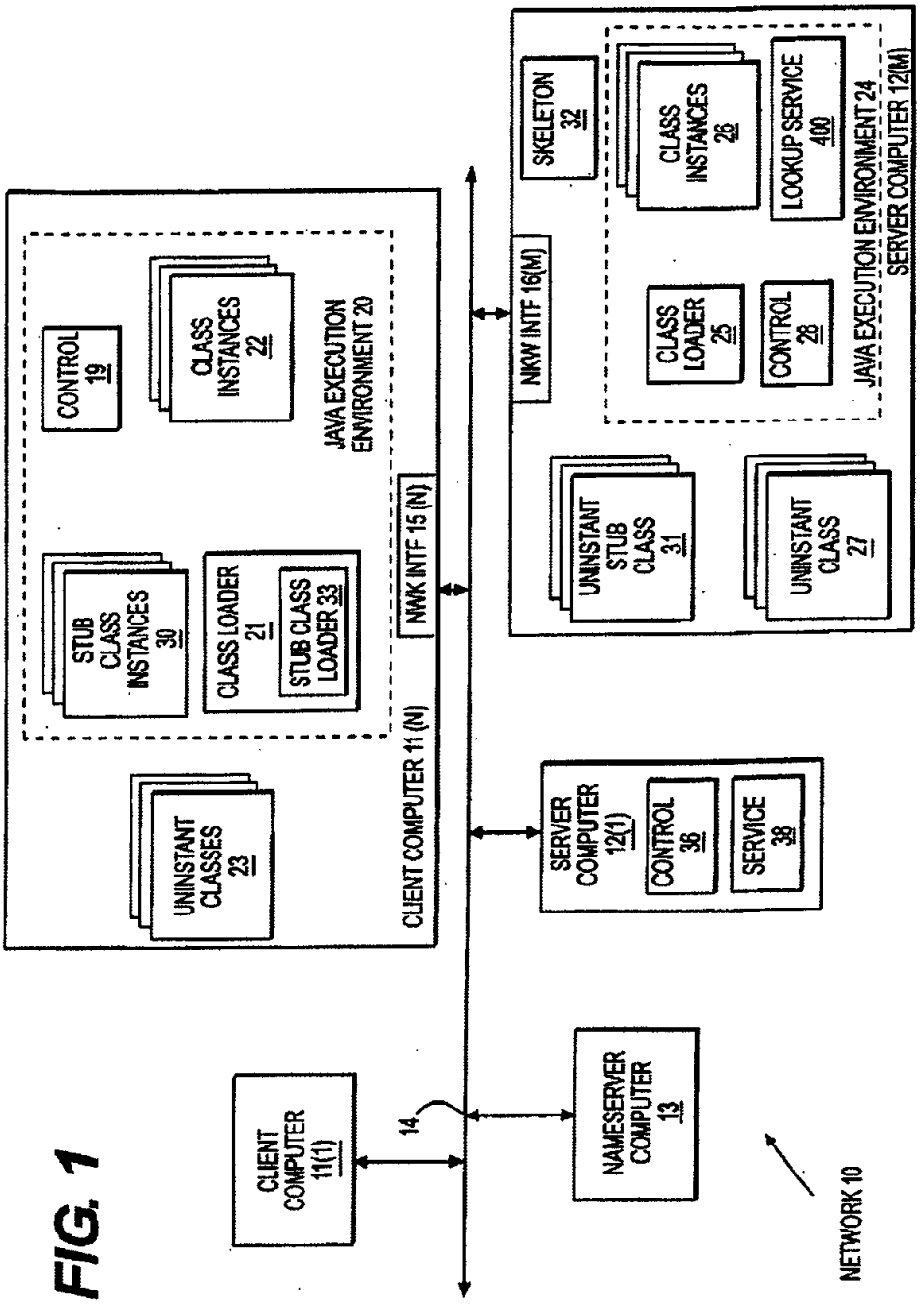

CLIENT COMPUTER 11 (N)
CONTROL 19
CLASS INSTANCES 22
JAVA EXECUTION ENVIRONMENT 20
STUB CLASS INSTANCES 30
CLASS LOADER 21
STUB CLASS LOADER 33
NWK INTF 15 (N)
UNINSTANT CLASSES 23
NWK INTF 16(M)
SKELETON 32
CLASS INSTANCES 26
LOOKUP SERVICE 400
CLASS LOADER 25
CONTROL 28
JAVA EXECUTION ENVIRONMENT 24
SERVER COMPUTER 12(M)
UNINSTANT STUB CLASS 31
UNINSTANT CLASS 27
SERVER COMPUTER 12(1)
CONTROL 36
SERVICE 39
CLIENT COMPUTER 11(1)
14
NAMESERVER COMPUTER 13
NETWORK 10

FIG. 1

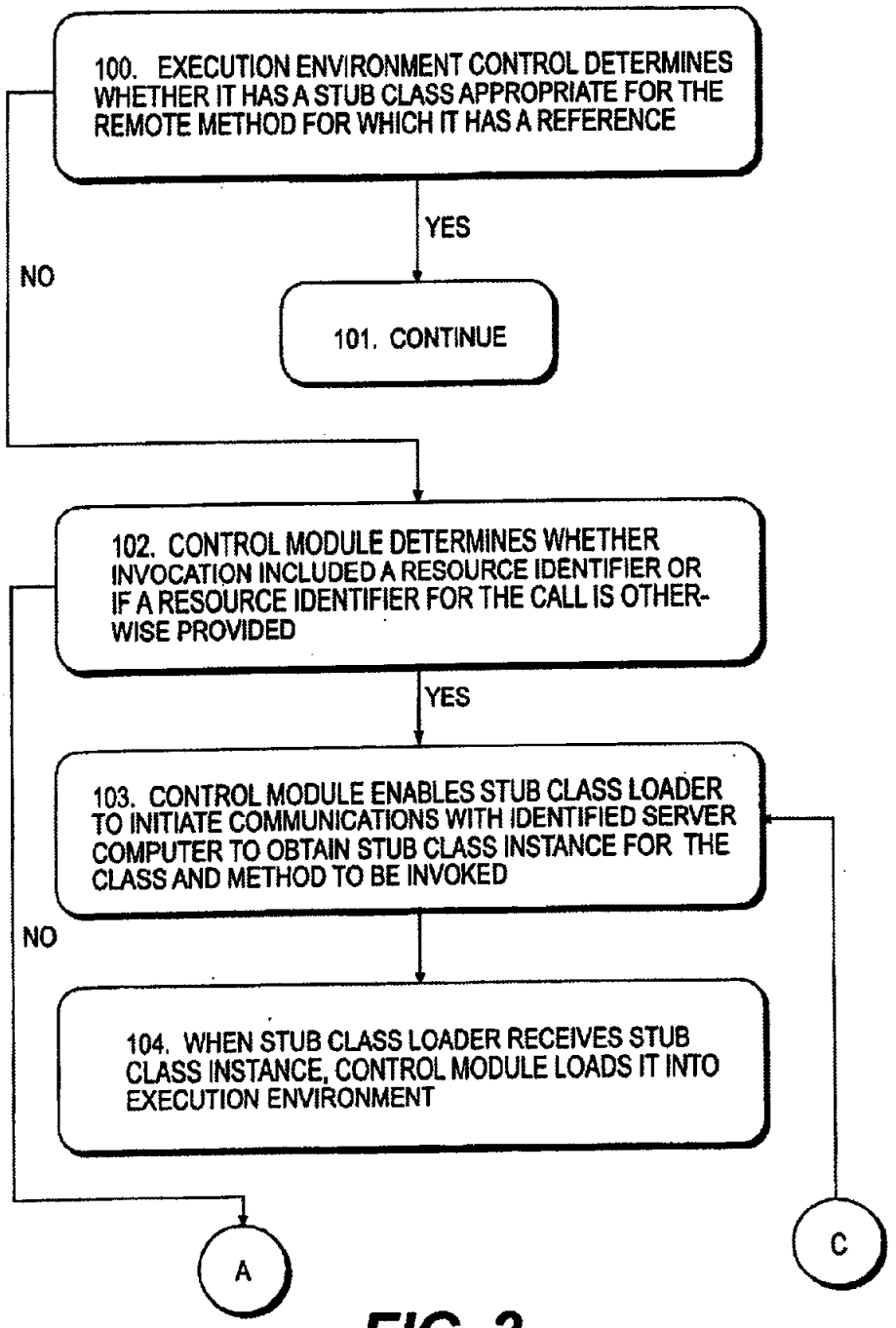

100. EXECUTION ENVIRONMENT CONTROL DETERMINES WHETHER IT HAS A STUB CLASS APPROPRIATE FOR THE REMOTE METHOD FOR WHICH IT HAS A REFERENCE
YES
NO
101. CONTINUE
102. CONTROL MODULE DETERMINES WHETHER INVOCATION INCLUDED A RESOURCE IDENTIFIER OR IF A RESOURCE IDENTIFIER FOR THE CALL IS OTHERWISE PROVIDED
YES
103. CONTROL MODULE ENABLES STUB CLASS LOADER TO INITIATE COMMUNICATIONS WITH IDENTIFIED SERVER COMPUTER TO OBTAIN STUB CLASS INSTANCE FOR THE CLASS AND METHOD TO BE INVOKED
NO
104. WHEN STUB CLASS LOADER RECEIVES STUB CLASS INSTANCE, CONTROL MODULE LOADS IT INTO EXECUTION ENVIRONMENT
A
C

FIG. 2

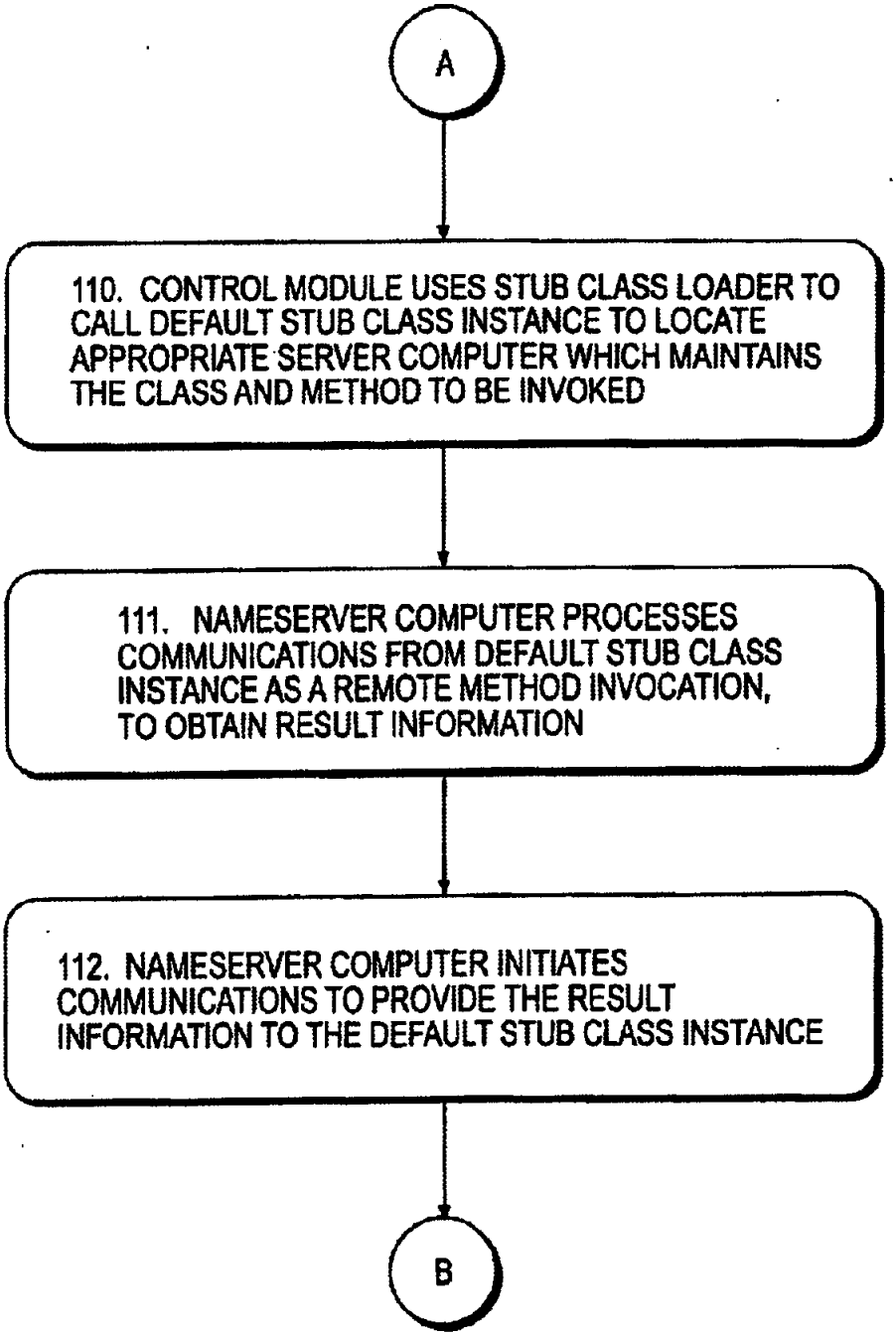

A
110. CONTROL MODULE USES STUB CLASS LOADER TO CALL DEFAULT STUB CLASS INSTANCE TO LOCATE APPROPRIATE SERVER COMPUTER WHICH MAINTAINS THE CLASS AND METHOD TO BE INVOKED
111. NAMESERVER COMPUTER PROCESSES COMMUNICATIONS FROM DEFAULT STUB CLASS INSTANCE AS A REMOTE METHOD INVOCATION, TO OBTAIN RESULT INFORMATION
112. NAMESERVER COMPUTER INITIATES COMMUNICATIONS TO PROVIDE THE RESULT INFORMATION TO THE DEFAULT STUB CLASS INSTANCE
B

FIG. 2 (CONT. A)

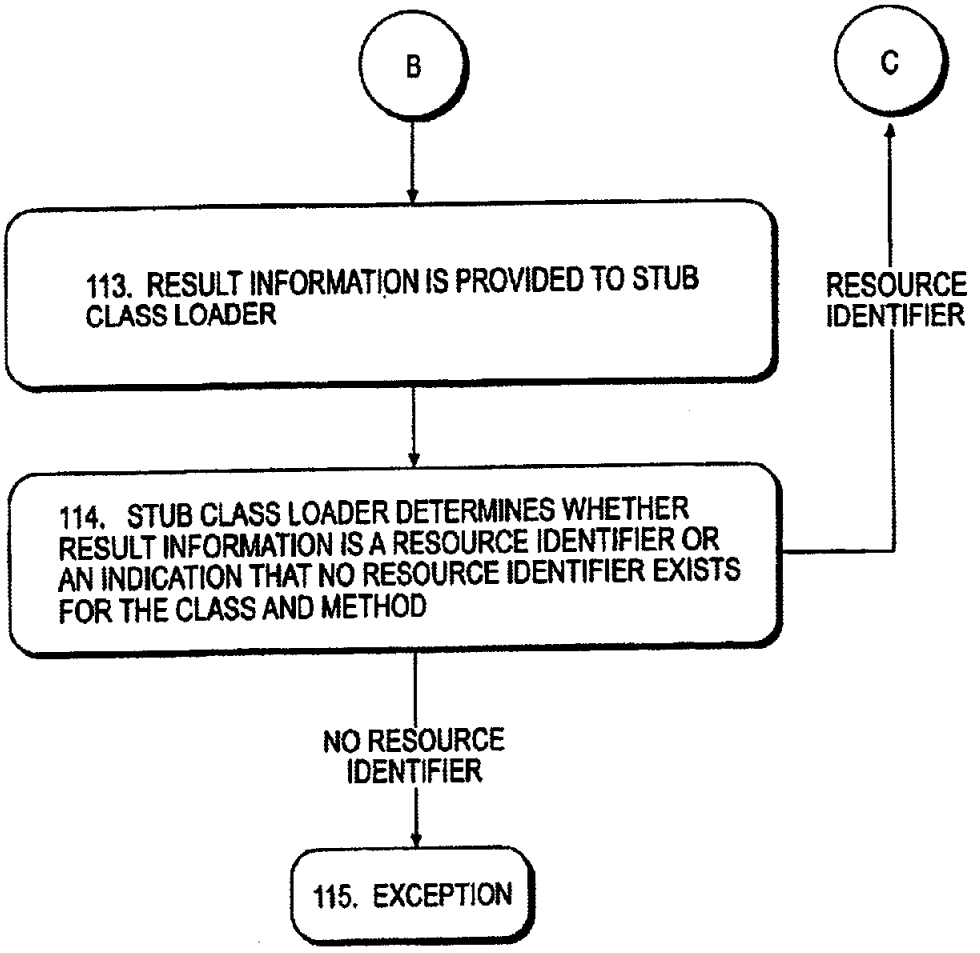

B
C
113. RESULT INFORMATION IS PROVIDED TO STUB CLASS LOADER
RESOURCE IDENTIFIER
114. STUB CLASS LOADER DETERMINES WHETHER RESULT INFORMATION IS A RESOURCE IDENTIFIER OR AN INDICATION THAT NO RESOURCE IDENTIFIER EXISTS FOR THE CLASS AND METHOD
NO RESOURCE IDENTIFIER
115. EXCEPTION

FIG. 2 (CONT. B)

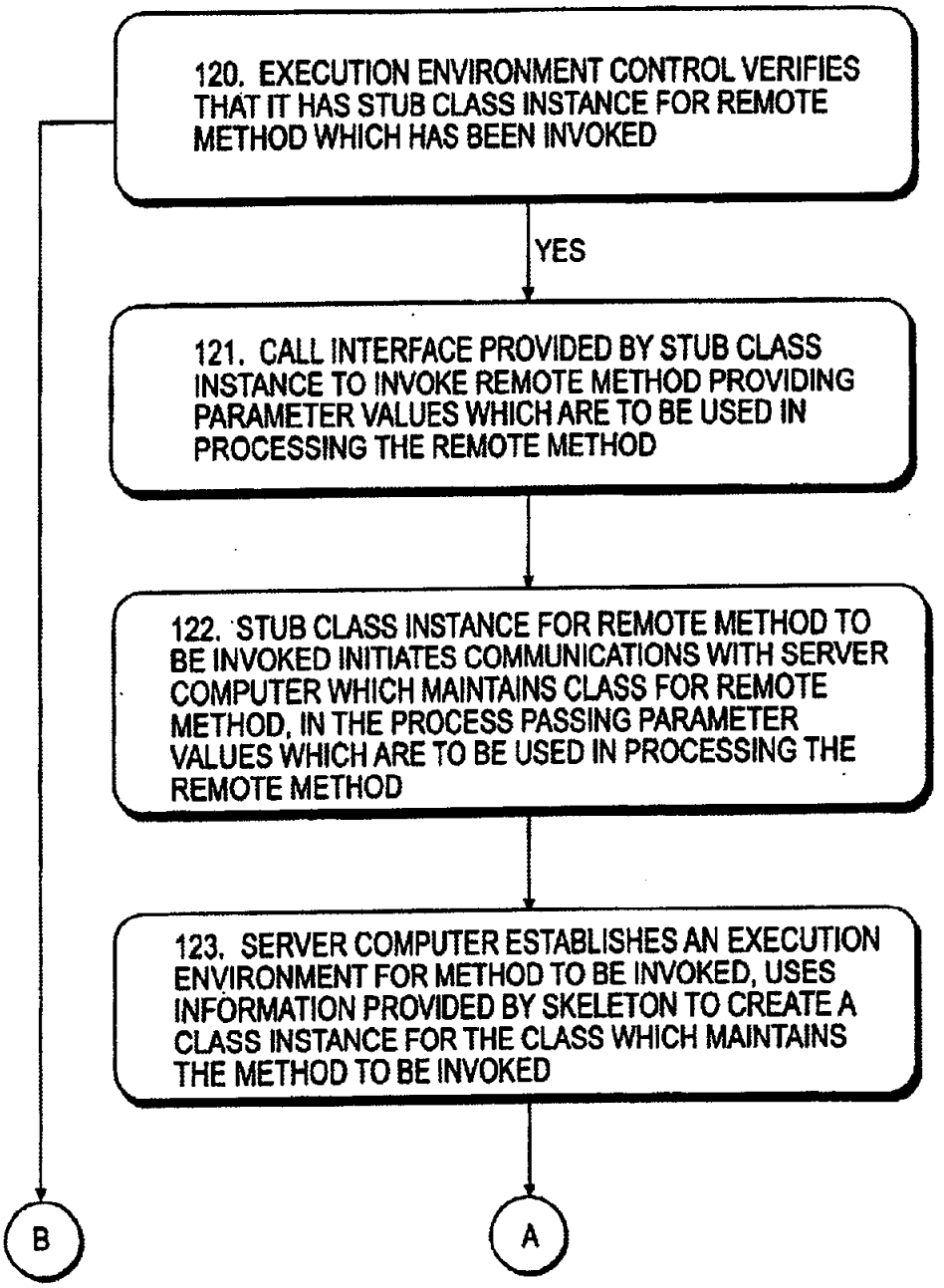

120. EXECUTION ENVIRONMENT CONTROL VERIFIES THAT IT HAS STUB CLASS INSTANCE FOR REMOTE METHOD WHICH HAS BEEN INVOKED
YES
121. CALL INTERFACE PROVIDED BY STUB CLASS INSTANCE TO INVOKE REMOTE METHOD PROVIDING PARAMETER VALUES WHICH ARE TO BE USED IN PROCESSING THE REMOTE METHOD
122. STUB CLASS INSTANCE FOR REMOTE METHOD TO BE INVOKED INITIATES COMMUNICATIONS WITH SERVER COMPUTER WHICH MAINTAINS CLASS FOR REMOTE METHOD, IN THE PROCESS PASSING PARAMETER VALUES WHICH ARE TO BE USED IN PROCESSING THE REMOTE METHOD
123. SERVER COMPUTER ESTABLISHES AN EXECUTION ENVIRONMENT FOR METHOD TO BE INVOKED, USES INFORMATION PROVIDED BY SKELETON TO CREATE A CLASS INSTANCE FOR THE CLASS WHICH MAINTAINS THE METHOD TO BE INVOKED
B
A

FIG. 3

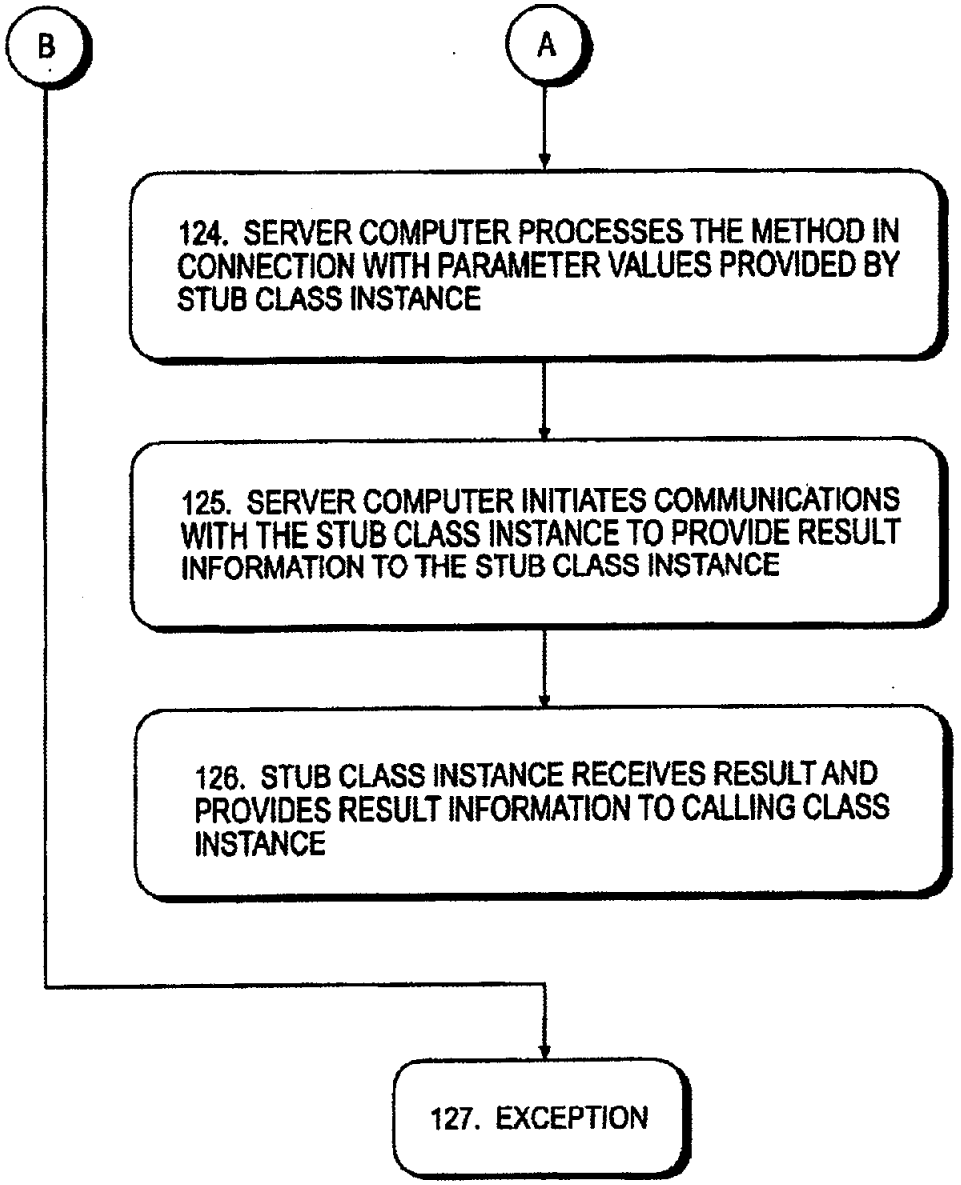

B
A
124. SERVER COMPUTER PROCESSES THE METHOD IN CONNECTION WITH PARAMETER VALUES PROVIDED BY STUB CLASS INSTANCE
125. SERVER COMPUTER INITIATES COMMUNICATIONS WITH THE STUB CLASS INSTANCE TO PROVIDE RESULT INFORMATION TO THE STUB CLASS INSTANCE
126. STUB CLASS INSTANCE RECEIVES RESULT AND PROVIDES RESULT INFORMATION TO CALLING CLASS INSTANCE
127. EXCEPTION

FIG. 3 (CONT. A)

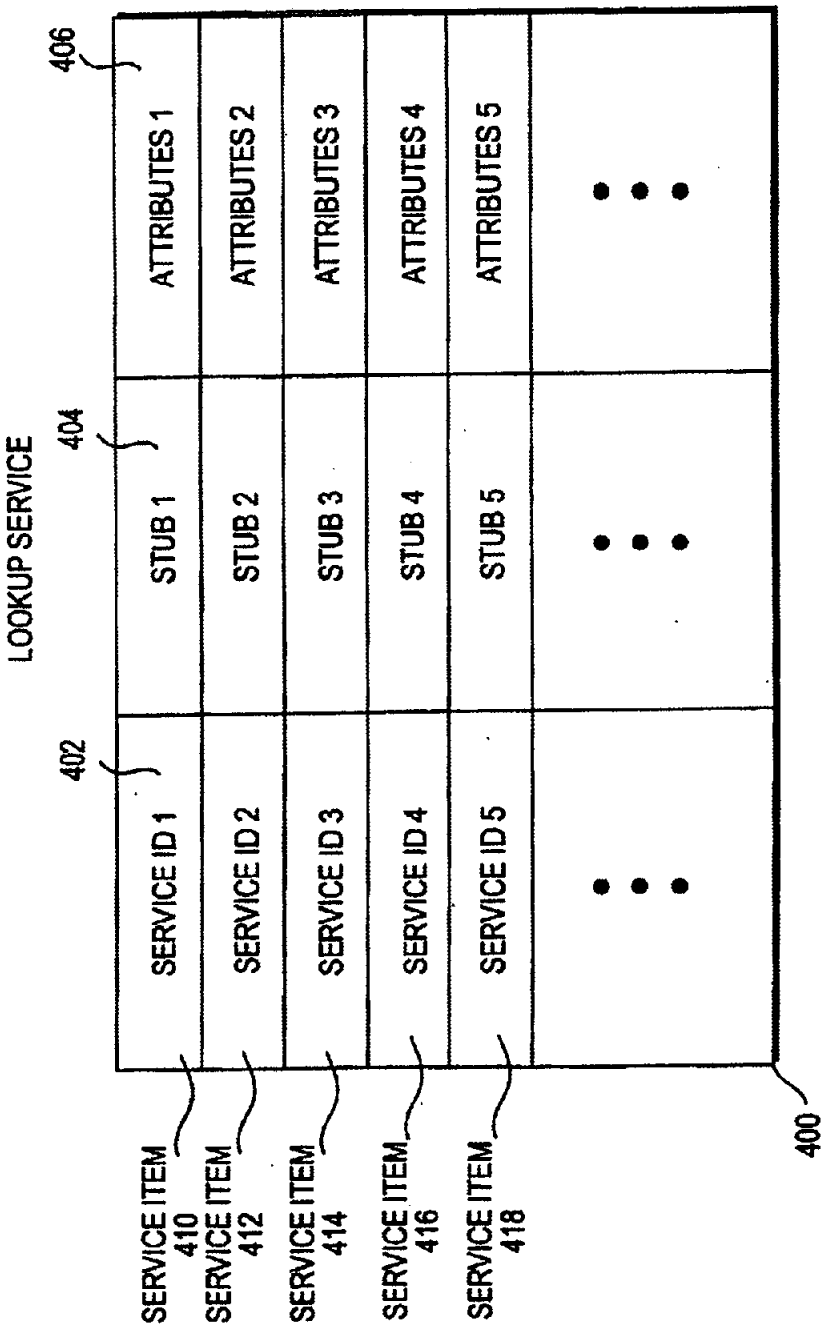

LOOKUP SERVICE
406
404
402
ATTRIBUTES 1
ATTRIBUTES 2
ATTRIBUTES 3
ATTRIBUTES 4
ATTRIBUTES 5
STUB 1
STUB 2
STUB 3
STUB 4
STUB 5
SERVICE ID 1
SERVICE ID 2
SERVICE ID 3
SERVICE ID 4
SERVICE ID 5
SERVICE ITEM 410
SERVICE ITEM 412
SERVICE ITEM 414
SERVICE ITEM 416
SERVICE ITEM 418
400

FIG. 4

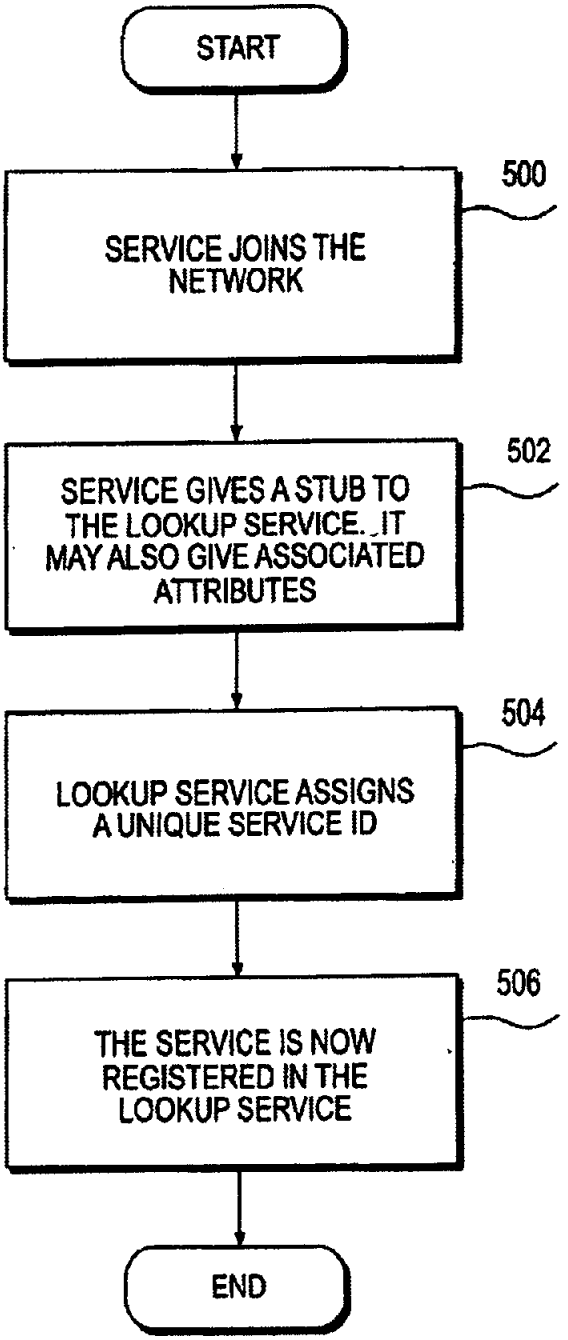

START
500
SERVICE JOINS THE NETWORK
502
SERVICE GIVES A STUB TO THE LOOKUP SERVICE. IT MAY ALSO GIVE ASSOCIATED ATTRIBUTES
504
LOOKUP SERVICE ASSIGNS A UNIQUE SERVICE ID
506
THE SERVICE IS NOW REGISTERED IN THE LOOKUP SERVICE
END

FIG. 5

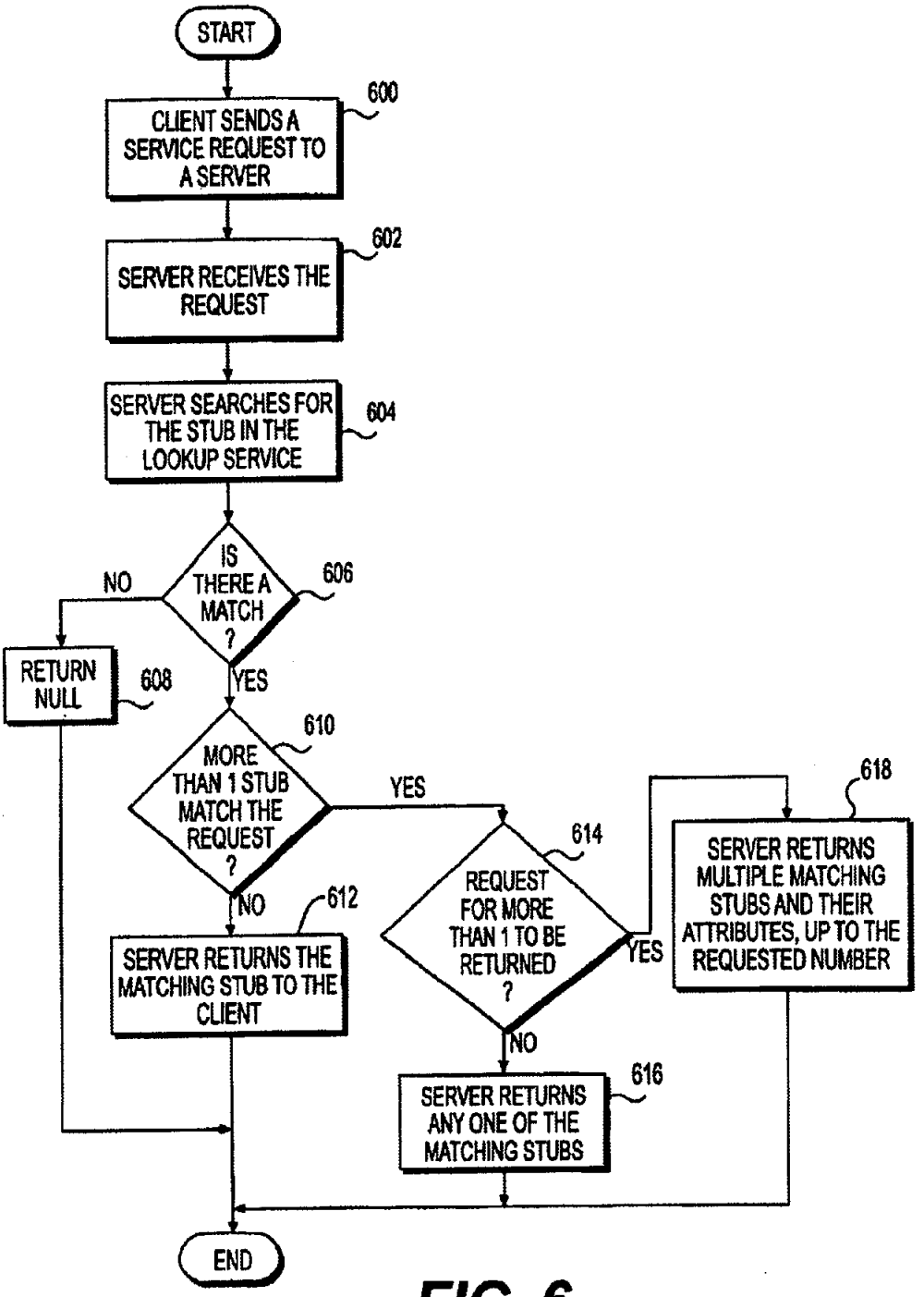

START
CLIENT SENDS A SERVICE REQUEST TO A SERVER
600
SERVER RECEIVES THE REQUEST
602
SERVER SEARCHES FOR THE STUB IN THE LOOKUP SERVICE
604
IS THERE A MATCH ?
606
NO
YES
RETURN NULL
608
MORE THAN 1 STUB MATCH THE REQUEST ?
610
YES
NO
SERVER RETURNS THE MATCHING STUB TO THE CLIENT
612
REQUEST FOR MORE THAN 1 TO BE RETURNED ?
614
YES
NO
SERVER RETURNS MULTIPLE MATCHING STUBS AND THEIR ATTRIBUTES, UP TO THE REQUESTED NUMBER
618
SERVER RETURNS ANY ONE OF THE MATCHING STUBS
616
END

FIG. 6

METHOD AND SYSTEM FOR FACILITATING ACCESS TO A LOOKUP SERVICE

RELATED APPLICATIONS

This application is a Continuation-In-Part of U.S. patent application Ser. No. 08/636,706 filed on Apr. 23, 1996, which is incorporated herein by reference.

The following applications are relied upon and are hereby incorporated by reference in this application.

Provisional U.S. Patent Application No. 60/078,048, entitled "Distributed Computing System," filed on Feb. 26, 1998.

U.S. patent application Ser. No. 09/044,923, entitled "Method and System for Leasing Storage," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,838, entitled "Method, Apparatus, and Product for Leasing of Delegation Certificates in a Distributed System," and filed on the same date herewith. U.S. patent application Ser. No. 09/044,834, entitled "Method, Apparatus and Product for Leasing of Group Membership in a Distributed System," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,916, entitled "Leasing for Failure Detection," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,933, entitled "Method for Transporting Behavior in Event Based System," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,919, entitled "Deferred Reconstruction of Objects and Remote Loading for Event Notification in a Distributed System," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,938, entitled "Methods and Apparatus for Remote Method Invocation," and filed on the same date herewith.

U.S. patent application Ser. No. 90/045,652, entitled "Method and System for Deterministic Hashes to Identify Remote Methods," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,790, entitled "Method and Apparatus for Determining Status of Remote Objects in a Distributed System," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,930, entitled "Downloadable Smart Proxies for Performing Processing Associated with a Remote Procedure Call in a Distributed System," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,917, entitled "Suspension and Continuation of Remote Methods," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,835, entitled "Method and System for Multi-Entry and Multi-Template Matching in a Database," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,839, entitled "Method and System for In-Place Modifications in a Database," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,945, entitled "Method and System for Typesafe Attribute Matching in a Database," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,931, entitled "Dynamic Lookup Service in a Distributed System," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,939, entitled "Apparatus and Method for Providing Downloadable Code for Use in Communicating with a Device in a Distributed System," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,932, entitled "Apparatus and Method for Dynamically Verifying Information in a Distributed System," and filed on the same date herewith.

U.S. patent application Ser. No. 09/030,840, entitled "Method and Apparatus for Dynamic Distributed Computing Over a Network," and filed on Feb. 26, 1998.

U.S. patent application Ser. No. 09/044,936, entitled "An Interactive Design Tool for Persistent Shared Memory Spaces," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,934, entitled "Polymorphic Token-Based Control," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,915, entitled "Stack-Based Access Control," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,944, entitled "Stack-Based Security Requirements," and filed on the same date herewith.

U.S. patent application Ser. No. 09/044,837, entitled "Per-Method Designation of Security Requirements," and filed on the same date herewith.

FIELD OF THE INVENTION

The present invention relates generally to data processing systems and, more particularly, to a method and system for facilitating access to a lookup service.

BACKGROUND OF THE INVENTION

In modern "enterprise" computing, a number of personal computers, workstations, and other devices such as mass storage subsystems, network printers and interfaces to the public telephony system, are typically interconnected in one or more computer networks. The personal computers and workstations are used by individual users to perform processing in connection with data and programs that may be stored in the network mass storage subsystems. In such an arrangement, the personal computers/workstations, operating as clients, typically download the data and programs from the network mass storage subsystems for processing. In addition, the personal computers or workstations will enable processed data to be uploaded to the network mass storage subsystems for storage, to a network printer for printing, to the telephony interface for transmission over the public telephony system, or the like. In such an arrangement, the network mass storage subsystems, network printers and telephony interface operate as servers, since they are available to service requests from all of the clients in the network. By organizing the network in such a manner, the servers are readily available for use by all of the personal computers/workstations in the network. Such a network may be spread over a fairly wide area, with the personal computers/workstations being interconnected by communication links such as electrical wires or optic fibers.

In addition to downloading information from servers for processing, a client, while processing a program, can remotely initiate processing by a server computer of particular routines and procedures (generally "procedures"), in connection with certain "parameter" information provided by the client. After the server has processed the procedure, it will provide results of its processing to the client, which the client may thereafter use in its processing operations.

Typically in such "remote procedure calls" the program will make use of a local "stub" which, when called, transfers the request to the server which implements the particular procedure, receives the results and provides them to the program. Conventionally, the stub must be compiled with the program, in which case the information needed to call the remote procedure must be determined at compile time, rather than at the time the program is run. Since the stub available to the client's programs is static, it may be at best the closest that can be determined should be provided for the program when it (the program) is compiled. Accordingly, errors and inefficiencies can develop due to mismatches between the stub that is provided to a program and the requirements of the remote procedure that is called when the program is run.

SUMMARY OF THE INVENTION

A new and improved system and method for facilitating the obtaining and dynamic loading of a stub is provided to enable a program operating in one address space to remotely invoke processing of a method or procedure in another address space, so that the stub can be loaded by the program when it is run and needed, rather than being statically determined when the program is compiled. Indeed, the stub that is loaded can be obtained from the resource providing the remote method or procedure, and so it (the stub) can exactly define the invocation requirements of the remote method or procedure. Since the stub can be located and dynamically loaded while the program is being run, rather that being statically determined when the program is compiled, run-time errors and inefficiencies which may result from mis-matches between the stub that is provided and the requirements of the remote method or procedure that is invoked can be minimized. In an alternative embodiment of the present invention, the stub is obtained from a lookup service to provide access to a service defined in the lookup service.

In brief summary, methods and systems consistent with an alternative embodiment of the present invention facilitate access to a service via a lookup service. A lookup service defines a network's directory of services and stores references to these services. A client desiring use of a service on the network accesses the lookup service, which returns the stub information that facilitates the user's access of the service. The client uses the stub information to access the service.

BRIEF DESCRIPTION OF THE DRAWINGS

This invention is pointed out with particularity in the appended claims. The above and further advantages of this invention may be better understood by referring to the following description taken in conjunction with the accompanying drawings, in which:

FIG. **1** is a function block diagram of a computer network including an arrangement constructed in accordance with the present invention for facilitating the obtaining, dynamic loading and use of "stub" information to enable a program operating in one address space to invoke processing of a remote method or procedure in another address space;

FIGS. **2** and **3** are flow charts depicting the operations performed by the arrangement depicted in FIG. **1**, with FIG. **2** depicting operations performed in connection with obtaining and dynamic loading of the stub information and FIG. **3** depicting operations performed in connection with use of the stub information to invoke processing of the remote method or procedure.

FIG. **4** is a diagram illustrating a lookup service consistent with the present invention;

FIG. **5** is a flowchart illustrating a method of adding a stub to the lookup service consistent with the present invention.

FIG. **6** is a flowchart illustrating a method for retrieving a stub from a lookup service by systems consistent with the present invention.

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

FIG. **1** is a schematic diagram of a computer network **10** including an arrangement for facilitating dynamic loading of "stub" information to enable a program operating in one address space to remotely invoke processing of a method or procedure in another address space, where this method or procedure represents a network service. With reference to FIG. **1**, computer network **10** includes a plurality of client computers **11**(1) through **11**(N) (generally identified by reference numeral **11**(*n*)), a plurality of server computers **12**(1) through **12**(M) (generally identified by reference numeral **12**(*m*)), all of which are interconnected by a network represented by a communication link **14**. In addition, the network **10** may include at least one nameserver computer **13**, which may also be connected to communication link **14**, whose purpose will be described below. As is conventional, at least some of the client computers **11**(*n*) are in the form of personal computers or computer workstations, each of which typically includes a system unit, a video display unit and operator input devices such as a keyboard and mouse (all of which are not separately shown). The server computers **12**(*m*) and nameserver computer **13** also typically include a system unit (also not separately shown), and may also include a video display unit and operator input devices.

The client computers **11**(*n*), server computers **12**(*m*) and nameserver computer **13** are all of the conventional stored-program computer architecture. A system unit generally includes processing, memory, mass storage devices such as disk and/or tape storage elements and other elements (not separately shown), including network interface devices **15**(*n*), **16**(*m*) for interfacing the respective computer to the communication link **14**. The video display unit permits the computer to display processed data and processing status to the operator, and an operator input device enables the operator to input data and control processing by the computer. The computers **11**(*n*) and **12**(*m*) and **13** transfer information, in the form of messages, through their respective network interface devices **15**(*n*), **16**(*m*) among each other over the communication link **14**.

In one embodiment, the network **10** is organized in a "client-server" configuration, in which one or more computers, shown in FIG. **1** as computers **12**(*m*), operate as servers, and the other computers, shown in FIG. **1** as computers **11**(*n*) operate as clients. In one aspect, one or more of the server computers **12**(*m*) may, as "file servers," include large-capacity mass storage devices which can store copies of programs and data which are available for retrieval by the client computers over the communication link **14** for use in their processing operations. From time to time, a client computer **11**(*n*) may also store data on the server computer **12**, which may be later retrieved by it (the client computer that stored the data) or other client computers for use in their processing operations. In addition, one or more of the server computers **12**(*m*) may, as "compute servers," perform certain processing operations in response to a remote request therefor from a client computer **11**(*n*), and

return the results of the processing to the requesting client computer **11**(*n*) for use by them (that is, the requesting client computers **11**(*n*)) in their subsequent processing. In either case, the server computers may be generally similar to the client computers **11**(*n*), including a system unit, video display unit and operator input devices and may be usable by an operator for data processing operations in a manner similar to a client computer. Alternatively, at least some of the server computers may include only processing, memory, mass storage and network interface elements for receiving and processing retrieval, storage or remote processing requests from the client computers, and generating responses thereto. It will be appreciated a client computer **11**(*n*) may also perform operations described herein as being performed by a server computer **12**(*m*), and similarly a server computer **12**(*m*) may also perform operations described herein as being performed by a client computer **11**(*n*).

The network represented by communication link **14** may comprise any of a number of types of networks over which client computers **11**(*n*), server computers **12**(*m*) and nameserver computers **13** may communicate, including, for example, local area networks (LANs) and wide area networks (WANs) which are typically maintained within individual enterprises, the public telephony system, the Internet, and other networks, which may transfer digital data among the various computers. The network may be implemented using any of a number of communication media, including, for example, wires, optical fibers, radio links, and/or other media for carrying signals representing information among the various computers depicted in FIG. **1**. As noted above, each of the computers typically includes a network interface which connects the respective computer to the communications link **14** and allows it to transmit and receive information thereover.

Systems consistent with the present invention facilitate the obtaining and dynamic loading of "stub" information to enable a program operating in one address space to invoke processing of a remote method or procedure in another address space, which may be located on the same computer as the invoking program or on a different computer. Reference will be made to programs provided in the Java™ programming language, as described in James Gosling, Bill Joy, Guy Steele, "The Java™ Language Specification", Addison-Wesley, 1996, (hereinafter referred to as the "Java language specification"), incorporated herein by reference, which are processed in connection with an execution environment which is provided by a Java virtual machine. The Java virtual machine, in turn, is specified in the Lindholm and Yellin, "The Java Virtual Machine Specification", Addison-Wesley, 1996, incorporated herein by reference. As described in the Java language specification, programs in the Java programming language define "classes" and "interfaces." Classes are used to define one or more methods or procedures, each of which may be invoked by reference to an interface. A class may be associated with and extend a "super-class," and in that regard will incorporate all of the interfaces and methods of the super-class, and may also include additional interfaces and/or methods. A class may also have one or more sub-classes (and thus will comprise a super-class of each of its sub-classes), with each sub-class incorporating and possibly extending their respective super-classes.

An interface provides a mechanism by which a set of methods may be declared. In that connection, an interface identifies each method that is declared by the interface by, for example, a name, identities the data type(s) of argument (s) that are to be provided for the method, the data type(s) of return values that are to be returned by the method, and identifiers for exceptions which can be thrown during processing of the method. A class may indicate that it implements a particular interface, and in that connection will include the program code which will be used in processing all of the methods which are declared in the interface. In addition, different classes may indicate that they implement the same interface, and each will have program code which will be used in processing all of the methods which are declared in the interface, but the program code provided in each class for use in processing the methods may differ from the program code provided in the other classes which is used in processing the same methods; thus, an interface provides a mechanism by which a set of methods can be declared without providing an indication of the procedure which will be used in processing any of the methods. An interface may be declared independently of the particular class which implements the method or methods which can be invoked using the interface. In that regard, a class that invokes the method and a class that actually implements the method will not need to share a common super-class.

During processing of a Java program, as described in the Java virtual machine specification, a client computer **11**(*n*) provides an execution environment **20** for interpreting the Java program. The Java virtual machine includes a class loader **21** that, under control of a control module **19**, can dynamically link instances of classes, generally identified in FIG. **1** by reference numeral **22**, into the running program's execution environment while the program is being executed. In that operation, the control module **19** effectively enables the class loader to retrieve uninstantiated classes, which generally identified by reference numeral **23**, instantiate them and link them as class instances **22** into the execution environment's address space at the Java program's run time as the methods which the respective classes **23** implement are called. In addition, the class loader **21** can discard ones of the class instances **22** when they are not needed or to conserve memory. It will be appreciated that, if a class instance **22** has been discarded, it may be reloaded by the class loader **21** at a later point if it is then needed.

Systems consistent with the present invention provide an arrangement which facilitates the remote invocation, by a program executing in an execution environment **20** by a client computer **11**(*n*), of methods implemented by classes on a server computer **12**(*m*). In executing a method, the server computer **12**(*m*) will also provide an execution environment **24** for processing, under control of a control module **28**, the Java method. In that operation, the Java virtual machine which provides the execution environment **24** includes a class loader **25** (which may be similar to the class loader **21**) that, under control of the control module **28**, can dynamically link an instance of the class **26**, to enable the method to be processed in the execution environment **24**, and instances of other classes (also generally represented by reference numeral **26**) which may be needed to process the remotely-invoked method. In that operation, the control module **28** effectively enables the class loader **25** to retrieve an uninstantiated class for the method to be invoked, from a plurality of uninstantiated classes which are generally identified by reference numeral **27**, instantiate it (that is, the uninstantiated class which provides the method to be invoked) and link it as a class instance **26** into the execution environment. In addition, the class loader **25** can discard the class instances **26** when processing of the method has terminated. It will be appreciated that, if class instances **26** has been discarded, it may be reloaded by the class loader **25** at a later point if it is then needed.

The structure of nameserver computer **13**, if provided, is generally similar to that of the server computer **12**(*m*), and will not be separately described.

To facilitate remote invocation of a method, the control module **19** of the client computer's execution environment **20** makes use of one or more stub class instances generally identified by reference numeral **30** which are provided as part of the execution environment **20** in which the various class instances **22**, including the class instance which is invoking the remote method, are being processed. Each stub class instance **30** is an instance of an uninstantiated stub class **31**, which the server computer **12**(*m*) may maintain for the various class instances **26** and uninstantiated classes **27** which the server computer **12**(*m*) has "exported," that is, which the server computer **12**(*m*) makes available to client computers **11**(*n*) for use in remote invocation of methods provided thereby. An uninstantiated stub class **31** includes declarations for the complete set of interfaces for the particular remote uninstantiated class **27** which implements the remote method to be invoked, and also provides or invokes methods which facilitate accessing of the remote method(s) which are implemented by the remote class. The uninstantiated stub class **31**, when it is instantiated and provided to the execution environment **20** of the client computer **11**(*n*) as a stub class instance **30**, effectively provides the information which is needed by the control module **19** of the execution environment **20** of the invoking Java program, so that, when a remote method that is implemented by its associated class is invoked by a Java program running in a particular execution environment, the remote method will be processed and the return value(s) provided to the invoking Java program. In one embodiment, the arrangement by which the stub class instance may be provided to the execution environment **20** is similar to that described in the aforementioned Waldo, et al., patent application.

In addition, the server computer **12**(*m*) provides a skeleton **32**, which identifies the particular classes and methods which have been exported by the server computer **12**(*m*) and information as to how it (that is, the server computer **12**(*m*)) may load the respective classes and initiate processing of the particular methods provided thereby. Additionally, the server computer **12**(*m*) contains a lookup service **400** for registering services on a network. The lookup service **400** will be discussed below.

When a class instance invokes a remote method maintained by a server computer **12**(*m*), it will provide values for various parameters to the stub class instance **30** for the remote method, which values the remote method will use in its processing. If the remote method is implemented on the same computer as the invoking Java program, when the invoking Java program invokes a remote method, the computer may establish an execution environment, similar to the execution environment **20**, enable the execution environment's class loader to load and instantiate the class which implements the method as a class instance similar to class instances **22**, and process the remote method using values of parameters which are provided by the invoking class instance in the remote invocation. After processing of the method has been completed, the execution environment in which the remote method has been processed will provide the results to the stub class instance **30** for the remote method that was invoked, which, in turn, will provide to the particular class instance **22** which invoked the remote method.

Similar operations will be performed if client computer **11**(*n*) and server computer **12**(*m*) are implemented on different physical computers. In that case, in response to a remote invocation, the client computer **11**(*n*) that is processing the invoking class instance **22**, under control of the control module **19** for the execution environment **20** for the invoking class instance **22**, will use the appropriate stub class instance **30** to communicate over the network represented by the communication link **14** with the server computer **12**(*m*) which implements the remote method to enable it (that is, the server computer **12**(*m*)) to establish an execution environment **24** for the class which implements the remote method, and to use the class loader **25** to load an instance of the class as a class instance **26**. In addition, the client computer **11**(*n*), also using the appropriate stub class instance **30**, will provide any required parameter values to the server computer **12**(*m*) over the network **14**. Thereafter, the server computer **12**(*m*) will process the remote method using parameter values so provided, to generate result value (s) which are transferred over the network to the client computer **11**(*n*), in particular to the appropriate stub class instance **30**. The client computer **11**(*n*) will, after it receives the result value(s) from the network, provide them to the invoking class instance **22** for its processing.

In any case, when the control module **19** of the client computer's execution environment **20** determines that a reference to the remote object has been received, if it determines that the stub class instance **30** is not present when it receives the reference, it will attempt to obtain the stub class instance **30** from, for example, the server computer **12**(*m*) which implements the remote method, and enable the stub class instance **30** to be dynamically loaded in the execution environment **20** for the invoking class instance **22**. A reference to the remote object may be received, for example, either as a return value of another remote method invocation or as a parameter that is received during another remote method invocation. The stub class instance may be dynamically loaded into the execution environment in a manner similar to that used to load class instances **22** in the execution environment **20**. The execution environment **20** is provided with a stub class loader **33** which, under control of the control module **19**, will attempt to find and load the stub class instances **30** as required by the class instances **22** processed in the execution environment. The location of a particular server computer **12**(*m*) that maintains the class that implements a method to be invoked remotely may be included in the call from the invoking class instance or may be made known to the stub class loader **33** through another mechanism (not shown) maintained by the client computer **11**(*n*).

However, if the stub class loader **33** is not otherwise notified of which server computer **12**(*m*) maintains the class which implements a method which may be invoked remotely, it may use the nameserver computer **13** to provide that identification. The identification may comprise any identifier which may be used to identify a server computer **12**(*m*) or other resource which is available on the network **14** and to which the server computer **12**(*m*) can respond. Illustrative identifiers include, for example, a network address which identifies the server computer and/or resource, or, if the network **14** is or includes the Internet, an identifier to, for example, a World Wide Web resource which may provide the identification or a "uniform resource locator" ("URL") which provides a uniform mechanism for identifying resources that are available over the Internet. The server computer **12**(*m*) which implements the remote method, in response to a request from the client computer **11**(*n*) will provide stub class instance **30** which the client computer **11**(*n*) may load into the execution environment **20** to thereafter enable the remote invocation to be initiated.

As noted above, if the stub class loader **33** does not know which server computer **12**(*m*) implements the remote method which may be invoked (and thus does not know which computer is to provide the stub class code for the remote invocation), it may, under control of the control module **19**, obtain the identification from the nameserver computer **13**. In that operation, the stub class loader **33** may use a previously-provided default stub class which is provided for use in such cases. The default class stub, when used by the invoking Java program, enables the computer that is processing the invoking Java program to communicate with the nameserver computer **13** to obtain information which can be used in invoking the remote method. This operation is essentially the same as the invocation of a remote method to be processed by the nameserver computer **13**, with the remote method including a parameter identifying the class and method to be remotely invoked, and enabling the nameserver computer **13** to provide the identification of a server computer **12**(*m*) which can process the method to the requesting client computer **11**(*n*) and other information which may be helpful in communicating with the server computer **12**(*m*) and invoking the particular method. It will be appreciated that the nameserver computer **13** will maintain a table (not separately shown) of "exported" resources, that is, resources, such as classes and methods, that are available to client computers **11**(*n*) connected to the network **14**, and information, such as the identifications of the particular server computers **12**(*m*) which provide those resources, which will be useful to the client computers **11**(*n*) in making use of the exported resources.

It will be appreciated that the nameserver computer **13** may create and maintain the exported resource table in a number of ways that are known in the art. For example, the nameserver computer **13** may periodically broadcast requests for exported resource information over the network **14**, to which the various server computers **12**(*m*) which maintain exported resources may respond; in that case, the nameserver computer **13** may establish its exported resource table based on the responses from the server computers **12**(*m*). Alternatively, each of the various server computers **12**(*m*) which maintains an exported resource may periodically broadcast information as to the exported resources which it maintains, and the nameserver computer **13** can update its exported resource table based on the broadcasts from the server computer. In addition, the nameserver computer's exported resource table may be established by a system operator and may be fixed until he or she updates it.

In any case, the information provided by the nameserver computer **13** in response to a request initiated by the default stub would include such information as, for example, the identification of a computer **12**(*m*) which can provide a class which implements the remote method to be invoked, particular information which the computer (that is, the computer which implements the remote method) will require to provide the required stub class code, and the like. After receiving the information from the nameserver computer **13**, the computer **11**(*n*) that is processing the invoking Java program may, under control of the control module **19**, use the information communicate with the computer (that is, the computer which implements the remote method) to obtain the stub class, and may thereafter invoke the method as described above.

With this background, the operations performed by client computer **11**(*n*), server computer **12**(*m*) and, if necessary, nameserver **13** in connection with obtaining and dynamic loading of a stub class instance when a reference to a remote method is received will be described in connection with the flow chart depicted in FIG. **2**. In addition, operations performed by the client computer **11**(*n*) and server computer in connection with remote invocation of a method using the stub class instance will be described in connection with the flow chart depicted in FIG. **3**. With reference initially to FIG. **2**, the execution environment control module **19** will, when it receives a reference to a remote method, will initially determine whether an appropriate stub class instance is present in the execution environment **20** to facilitate invocation of the remote method (step **100**). If the control module **19** determines that such a stub class instance **30** for the remote method is present in the execution environment, it may continue other operations (step **101**). However, if the control module **19** determines in step **101** that such a stub class instance is not present in the execution environment **20** for the remote method, the control module **19** will use the stub class loader **33** to attempt to locate and load a stub class instance **30** for the class to process the remote method. In that case, the control module **19** will initially determine whether the invocation from the class instance **22** included a resource locator to identify the server computer **12**(*m*) or other resource which maintains the class for the method to be invoked, or whether it (that is, the control module **19**) or the stub class loader **33** otherwise are provided with such a resource locator (step **102**). If the control module **19** makes a positive determination in that step, it will sequence to step **103** to enable the stub class loader **33** to initiate communications with identified server computer **12**(*m*) to obtain a stub class instance for the class and method to be invoked (step **103**). When the stub class loader **33** receives the stub class instance **30** from the server computer **12**(*m*), it will load the stub class instance **30** into execution environment **20** for the class instance **22** which initiated the remote method invocation call in step **100** (step **104**). After the stub class instance **30** for the referenced remote method has been loaded in the execution environment, the method can be invoked as will be described below in connection with

Returning to step **102**, if the control module **19** determines that the invocation from the class instance **22** did not include a resource locator to identify the server computer **12**(*m*) or other resource which maintains the class for the method to be invoked, and further that it (that is, the control module **19**) or the stub class loader **33** is not otherwise provided with such a resource locator, a "class not found" exception may be indicated, at which point the control module **19** may call an exception handler. The exception handler may perform any of a number of recovery operations, including, for example, merely notifying the control module **19** that the remote method could not be located and allow it to determine subsequent operations.

Alternatively, the control module **19** may attempt to obtain a resource locator from the nameserver computer **13** or other resource provided by the network **14** (generally represented in FIG. **1** by the nameserver computer **13**), using a call, for example, a default stub class instance **30**. The call to the default stub class instance **30** will include an identification of the class and method to be invoked and the name of the nameserver computer **13**. Using the default stub class instance **30**, the control module **19** will enable the computer **11**(*n*) to initiate communications with nameserver computer **13** to obtain an identifier for a server computer **12**(*m*) which maintains the class and method to be invoked (step **110**). The communications from the default stub class instance **30** will essentially correspond to a remote method invocation, with the method enabling the nameserver computer to provide the identification for the server computer **12**(*m*), if one exists

associated with the class and method to be remotely invoked, or alternatively to provide an indication that no server computer **12**(*m*) is identified as being associated with the class and method. During the communications in step **110**, the default stub class interface **30** will provide, as a parameter value, the identification of class and method to be invoked.

In response to the communications from the default stub class instance **30**, the nameserver computer **13** will process the request as a remote method (step **111**), with the result information comprising the identification for the server computer **12**(*m*), if one exists that is associated with the class and method to be remotely invoked, or alternatively an indication that no server computer **12**(*m*) is identified as being associated with the class and method. After finishing the method, the nameserver computer **13** will initiate communications with the default stub class instance **30** to provide the result information to the default stub class instance **30** (step **112**).

After receipt of the result information from the nameserver computer **13**, the default stub class instance, under control of the control module **19**, will pass result information to the stub class loader **33** (step **113**). Thereafter, the stub class loader **33** determines whether the result information from the nameserver computer comprises the identification for the server computer **12**(*m*) or an indication that no server computer **12**(*m*) is identified as being associated with the class (step **114**). If the stub class loader **33** determines that the result information comprises the identification for the server computer **12**(*m*), it (that is, the stub class loader **33**) will return to step **103** to initiate communication with the identified server computer **12**(*m*) to obtain stub class instance for the class and method that may be invoked. On the other hand, if the stub class loader **33** determines in step **114** that the nameserver computer **13** had provided an indication that no server computer **12**(*m*) is identified as being associated with the class and method that may be invoked, the "class not found" exception may be indicated (step **115**) and an exception handler called as described above.

As noted above, the stub class instance **30** retrieved and loaded as described above in connection with FIG. **2** may be used in remote invocation of the method. Operations performed by the client computer **11**(*n*) in connection with remote invocation of the method will be described in connection with the flow chart in FIG. **3**. As depicted in FIG. **3**, when a class instance **22** invokes a method, the control module **19** may initially verify that a stub class instance **30** is present in the execution environment for remote method to be invoked (step **120**). If a positive determination is made in step **120**, the stub class instance **30** will be used for the remote invocation, and in the remote invocation will provide parameter values which are to be used in processing the remote method (step **121**). Thereafter, the stub class instance **30** for the remote method that may be invoked will be used to initiate communications with the server computer **12**(*m*) which maintains the class for the remote method (step **122**), in the process, the passing parameter values which are to be used in processing the remote method will be passed. It will be appreciated that, if the server computer **12**(*m*) which is to process the method is the same physical computer as the client computer **12**(*n*) which is invoking the method, the communications can be among execution environments which are being processed within the physical computer. On the other hand, if the server computer **12**(*m*) which is to process the method is a different physical computer from that of the client computer **11**(*n*) which is invoking the method, the communications will be through the client computer's and server computer's respective network interfaces **15**(*n*) and **16**(*m*) and over the network **14**.

In response to the communications from the stub class instance in step **122**, the server computer **12**(*m*), if necessary establishes an execution environment **24** for the class which maintains the method that may be invoked, and the uses the information provided by the skeleton **32** to create a class instance **26** for that class (step **123**). Thereafter, the server computer **12**(*m*), under control of the control module **28**, will process the method in connection with parameter values that were provided by stub class instance **30** (step **124**). After completing processing of the method, the server computer **12**(*m*), also under control of the control module **28**, will initiate communications with the client computer's stub class instance **30** to provide result information to the stub class instance (step **125**). In a manner similar to that described above in connection with step **102**, if the server computer **12**(*m*) which processed the method is the same physical computer as the client computer **11**(*n*) which invoked the method, the communications can be among execution environments **24** and **20** which are being processed within the physical computer. On the other hand, if the server computer **12**(*m*) which processed the method is a different physical computer from that of the client computer **11**(*n*) which is invoking the method, the communications will be through the server computer's and client computer's respective network interfaces **16**(*m*) and **15**(*n*) and over the network **14**. After the stub class instance **30** receives the result information from the server computer, it may provide result information to the class instance **22** which initiated the remote method invocation (step **126**), and that class instance **22** can continue processing under control of the control module **19**.

Returning to step **120**, if the control module **19** determines in that step that it does not have a stub class instance **30** that is appropriate for the remote method that may be invoked, it may at that point call an exception handler (step **127**) to perform selected error recovery operations.

Methods and systems consistent with the present invention provide a number of advantages. In particular, they provide a new system and method for facilitating dynamic loading of a stub which enables a program that is operating in one execution environment to remotely invoke processing of a method in another execution environment, so that the stub can be loaded by the program when it is run and needed. In systems in which stubs are compiled with the program, and thus are statically determined when the program is compiled, they (the stubs) may implement subsets of the actual set of remote interfaces which are supported by the remote references that are received by the program, which can lead to errors and inefficiencies due to mismatches between the stub that is provided to a program and the requirements of the remote procedure that is called when the program is run. However, since, in the dynamic stub loading system and method, the stub that is loaded can be obtained from the particular resource which provides the remote method, it (the stub) can define the exact set of interfaces to be provided to the invoking program at run time, thereby obviating run-time incompatibilities which may result from mismatches between the stub that is provided and the requirements of the remote method that is invoked.

It will be appreciated that a number of modifications may be made to the arrangement as described above. For example, although the execution environment **20** has been described as obtaining and loading stub class instances to facilitate invocation of remote methods when references to

the remote methods are received, it will be appreciated that stub class instances may instead be obtained and loaded when the remote methods are initially invoked. Obtaining and loading of the stub class instance for a remote method when a reference thereto is received will have the advantages that (i) the stub class instance will be present in the execution environment when the remote method is actually invoked, and (ii) if the appropriate stub class instance can not be located, the program or an operator may be notified at an early time. On the other hand, obtaining and loading of the stub class instance for a remote method when the method is to be invoked may result in a delay of the invocation until the correct stub class instance can be found, if the method is in fact not invoked even if a reference to it is received the stub class instance may not need to be located and loaded.

It will be appreciated that a system in accordance with the invention can be constructed in whole or in part from special purpose hardware or a general purpose computer system, or any combination thereof, any portion of which may be controlled by a suitable program. Any program may in whole or in part comprise part of or be stored on the system in a conventional manner, or it may in whole or in part be provided to the system over a network or other mechanism for transferring information in a conventional manner. In addition, it will be appreciated that the system may be operated and/or otherwise controlled by means of information provided by an operator using operator input elements (not shown) which may be connected directly to the system or which may transfer the information to the system over a network or other mechanism for transferring information in a conventional manner.

ALTERNATIVE EMBODIMENT OF THE PRESENT INVENTION

Although an embodiment consistent with the present invention has been previously described that dynamically retrieves and loads stubs, an alternative embodiment also consistent with the present invention loads and retrieves objects in a lookup service, where the objects contain code (stub information) for facilitating communication with a particular service or the objects contain code that performs the service. Although the alternative embodiment is described below as downloading objects from the lookup service that represent stubs, the techniques described below are equally applicable to downloading objects that actually perform the services. A lookup service defines a network's directory of services and stores references to these services. A user desiring use of a service on the network accesses the lookup service, which returns the stub information that facilitates the user's access of the service.

The lookup service may contain a subset of all services available in the network, referred to as a "Djinn" as described in copending U.S. patent application Ser. No. 09/044,931, entitled "Dynamical Lookup Service in a Distributed System," assigned to a common assignee, filed on even date herewith, which has been previously incorporated by reference. A "Djinn" refers to a logical grouping of one or more of the services or resources that are provided by a network. Devices connected to the network may either dynamically add themselves to the Djinn or dynamically remove themselves from the Djinn. When added, a device provides zero or more of its services to the Djinn and may utilize all of the services currently provided by the Djinn. The services provided by the Djinn are defined by the lookup service, which provides a common way to both find and utilize the services for the Djinn.

The lookup service is a fundamental part of the infrastructure for a Djinn or other computer network offering a range of services. It is the primary means for programs to find services available within the Djinn and is the foundation for providing stubs through which users and administrators can discover and interact with services in the Djinn.

Reference will now be made to FIG. **4**, which depicts lookup service **400** in greater detail. Server computer **12**(*m*), also includes a lookup service **400**, further described below. The lookup service **400**, located on a server **12**(*m*) as shown in FIG. **1**, maintains a collection of "service items" **410–418**. Each service item **410–418** represents an instance of a service available within the Djinn, and each service item **410–418** contains a service ID **402** that uniquely identifies the service item, a stub **404** providing code that programs use to access the service, and a collection of attributes **406** that describe the service.

Upon registering a new service with the lookup service **400**, the lookup service gives the new service item **410** a unique service ID **402**, typically a number. This service ID **402** can later be used to access the specific service, avoiding unnecessary searching or locating several matching service items upon a query.

When a new service is created (e.g., when a new device is added to the Djinn), the service registers itself with the lookup service **400**, providing a stub **404** to be used by a client to access the service and an initial collection of attributes **406** associated with the service. For example, a printer might include attributes indicating speed (in pages per minute), resolution (in dots per inch), color, and whether duplex printing is supported. The lookup service administrator (not shown) might also add new attributes, such as the physical location of the service and common names for it. Additionally, if a service encounters some problem that needs administrative attention, such as a printer running out of toner, the service can add an attribute that indicates what the problem is. In one implementation consistent with the present invention, attributes are stored as multi-entries, and the addition, modification and deletion of attributes can be made using multi-templates and the techniques explained in co-pending U.S. patent application Ser. No. 09/044,839, entitled "Method and System for In-Place Modifications In A Database", previously incorporated herein.

An individual set of attributes is represented as an instance of a class, each attribute being a field of that class. An example of an attribute set for a printer is:

```
public class Printer {
    Integer ppm;        //  pages per minute
    Integer dpi;        //  resolution in dots per inch
    Boolean duplex;     //  supports two-sided printing
    Boolean color;      //  color or black-only
}
```

The class provides strong typing of both the set and the individual attributes.

The attributes **406** of service items **410–418** can also be represented as a set of sets of attributes. Attributes **406** of a service item **410** can contain multiple instances of the same class with different attribute values, as well as multiple instances of different classes. For example, the attributes **406** of a service item **410** might have multiple instances of a Name class, each giving the common name of the service in a different language, plus an instance of a Location class, a Type class, or various other service-specific classes. An example of some added attributes to describe the printer may be the Name, Type, or Location:

```
    public class Name implements Entry {
        String name;          //  the user-friendly name of the service
        String description;   //  free-form description of the service
        String language;      //  language (e.g., English, French) used
in the above
}
    public class Type implements Entry {
        String type;          //  the general type of service
        String vendor;        //  vendor of product that implements
the service
        String model;         //  model number/name of product
        String version;       //  version number of product
}
    public class Location implements Entry {
        Integer floor;        //  what floor the service is on
        String building;      //  what building it's in
        String room;          //  what room it's in
}
```

In this example, the attributes **406** for this service item **410** would be a set of attributes containing the Printer, Name, Type, and Location class instances, each class containing their own individual attributes. However, it should be noted that the scheme used for attributes is not constrained by these examples.

Programs (including other services) that need a particular type of service can use the lookup service **400** to find a stub that can be used to access the service. A match can be made based on the type of service as well as the specific attributes attached to the service. For example, a client could search for a printer by requesting a stub type corresponding to the service desired or by requesting certain attributes such as a specific location or printing speed. In one implementation consistent with the present invention, attributes are stored as multi-entries, and a match on attributes can be made using multi-templates, as explained in co-pending U.S. patent application Ser. No. 09/044,835, entitled "Method and System For Multi-Entry and Multi-Template Matching In A Database", previously incorporated herein.

Accessing a Lookup Service Employing Dynamic Stub Loading and Retrieval

Referring back to FIG. **4**, the stub **404** corresponding to a service is registered in the lookup service **400** and is used by the client computer **11**(*n*) to access the service methods remotely. This stub **404** may also be a "smart proxy." A smart proxy, code within which a stub is embedded, helps the client more efficiently implement the stub and the method to be remotely invoked. A smart proxy often performs some local computation for efficiency before or after it actually calls the stub. For example, a smart proxy may contain code to cache information, so if a client requested it again, instead of going back to the server to get the information, it may have cached the answer and be able to return it quickly. If the situation called for it, a smart proxy might also transform the parameters received from the client into other types and then send the transformed types. The smart proxy concept is further explained in co-pending U.S. patent application Ser. No. 09/044,930, entitled "Downloadable Smart Proxies for Performing Processing Associated with a Remote Procedure Call in a Distributed System," assigned to a common assignee, filed on even date herewith, which is hereby incorporated by reference.

FIG. **5** depicts a flowchart illustrating the steps used by systems consistent with the present invention for adding a service stub to the lookup service. When a device joins the network, it typically registers a service with the lookup service (step **500**). Upon registration with the lookup service **400**, the device supplies a stub **404** to the lookup service, and it may also give its associated attributes **406** to the lookup service (step **502**). In response, the lookup service **400** assigns a unique service ID **402**, typically a number as previously stated, to the service registered with the lookup service (step **504**). Once the device has supplied the lookup service **400** with a stub **404** and attributes **406**, and the lookup service has assigned a unique service ID **402**, the device has completed registration of the service with the lookup service (step **506**). After services are registered with the lookup service **400**, clients can use the lookup service to obtain stub information needed to access the registered services.

FIG. **6** depicts a flowchart illustrating steps used by systems and methods consistent with the present invention to download a service item from the lookup service. In one implementation, the client computer **11**(*n*) sends a request for a service to the server **12**(*m*) with the lookup service **400** (step **600**). The request originates from the remote method invocation of a class instance **22** on client computer **11**(*n*), and the requested service may reside on a remote server as the exemplary service **38** resides on server **12**(**1**). In one implementation consistent with the present invention, the client computer **11**(*n*) may request one or more services from the lookup service **400**. The client's request comes in the form of a specific service ID **402**, a type of stub **404**, or a set of attributes **406**, or any combination thereof (step **602**). In response to the request, control **19** directs the stub class loader **33** to locate the corresponding stub **404** from the server **12**(*m*). To do so, the control **19** enables the stub class loader **33** to initiate communication with the server **12**(*m*) to obtain a stub **404** for the service to be obtained.

Upon receipt of the request from the client computer **11**(*n*), the control **28** in the server **12**(*m*) searches the lookup service **400** for the stub **404** corresponding to the requested service (step **604**). If there are no matches found, the control **28** returns a null value (steps **606** and **608**). Otherwise, if it locates the stub **404** corresponding to the service that the client computer **11**(*n*) is attempting to access, the server **12**(*m*) returns the stub to the stub class loader **33** on the client computer (step **612**). If more than one stub was located matching the client's request (step **610**), in one embodiment consistent with the present invention, any one of the stubs is returned (step **616**). In another implementation where the client requests more than one service, the server **12**(*m*) returns the requested number of the stubs with their attributes (steps **614** and **618**).

When the stub **404** is received by the stub class loader **33**, the stub class loader loads it into the execution environment **20**. After it is loaded, the service **38** can be remotely invoked. The use of the stub information to invoke remote processing of the service **38** is performed in the same manner as previously discussed in connection with FIG. **3**.

Generally, the class instance **22** can use the stub **404** to access the service **38** on the server **12**(**1**). When the class instance **22** requires use of the service **38** corresponding to the returned stub **404**, control **19** verifies that the stub **404** is present in the execution environment **20**. If so, the class instance **22** can then use the stub **404** to initiate communications with the server **12**(**1**) that maintains the service **38**, and parameters will be passed to the service **38** for implementation.

This lookup service implementation is one application of the dynamic loading and retrieval of stub information to enable a program operating in one address space to invoke

processing of a procedure in another address space. This implementation of using the dynamic stub loading on the lookup service allows a client to receive stub information to facilitate use of that service directly. Unlike previous lookup services, the lookup service consistent with the present invention returns the code needed to access the service directly. Using the dynamic loading of stub information in this way allows the client to receive all the code necessary to facilitate use of the service on a remote server.

The foregoing description has been limited to a specific embodiment of the present invention. It will be apparent, however, that various variations and modifications may be made. It is the object of the appended claims to cover these and such other variations and modifications as come within the true spirit and scope of the invention.

What is claimed is:

1. A method in a data processing system for accessing network services associated with a lookup service, comprising the steps of:

receiving a request by the lookup service for access to one of the network services; and

returning, by the lookup service, executable code to facilitate the access of the one network service.

2. The method of claim **1**, further including the step of:

accessing the network service using the returned executable code.

3. The method of claim **1**, wherein the step of returning the executable code includes the step of:

returning a stub.

4. The method of claim **1**, wherein the step of returning the executable code includes the step of:

returning a smart proxy.

5. The method of claim **1**, wherein the receiving step includes the step of:

receiving the request to access the one network service such that the one network service is identified by a type of service.

6. The method of claim **1**, wherein the receiving step includes the step of:

receiving the request to access the one network service such that the one network service is identified by a service identification number.

7. The method of claim **1**, wherein the receiving step includes the step of:

receiving the request to access the one network service such that the one network service is identified by an attribute of the one network service.

8. The method of claim **1**, wherein the step of returning executable code includes the step of:

returning a plurality of stubs associated with a plurality of network services that match the request.

9. The method of claim **8**, wherein the step of returning a plurality of stubs includes the step of:

returning attributes associated with each stub.

10. A method in a data processing system having a lookup service with a plurality of services, the method comprising the steps of:

sending to the lookup service a request to access one of the services; and

receiving executable code, from the lookup service and responsive to the request, that facilitates use of the one service.

11. The method of claim **10**, further including the step of:

accessing the one service using the received executable code.

12. The method of claim **10**, wherein the step of receiving executable code includes the step of:

receiving a stub that facilitates access of the one service.

13. A distributed system with a plurality of network services, comprising:

a server computer with a lookup service having stubs for facilitating access to the network services; and

a client computer with a program that sends a request to the lookup service for one of the stubs corresponding to one of the network services, that receives the stub, and that accesses the one network service using the received stub.

14. A method in a data processing system having a first computer with a client and a second computer with a lookup service containing service stubs used for accessing associated services, the method comprising the steps of:

sending a request by the client to the lookup service identifying one of the associated services to be accessed;

receiving the request by the lookup service;

searching the lookup service for the identified service;

returning a service stub associated with the identified service to the client;

receiving the service stub by the client;

loading the service stub into an address space of the client to render the service stub available for use to invoke the identified service; and

using the stub by the client to access the identified service.

15. The method of claim **14**, wherein the sending step includes the step of:

sending a request for more than one service to be returned, and wherein the returning step further includes the step of:

returning more than one stub to the client in response to the request.

16. The method of claim **14**, wherein the searching step includes the step of:

returning a null value if no service is found matching the request.

17. The method of claim **10**, wherein the step of receiving executable code includes the step of:

receiving a smart proxy that facilitates access of the one service.

18. The distributed system of claim **13**, wherein the client computer utilizes a remote procedure call mechanism to receive the stub.

19. A system having a first computer with a client and a second computer with a lookup service containing service stubs used for accessing associated services, the system comprising:

means for sending a request by the client to the lookup service identifying one of the associated services to be accessed;

means for receiving the request by the lookup service;

means for searching the lookup service for the identified service;

means for returning a service stub associated with the identified service to the client;

means for receiving the service stub by the client;

means for loading the service stub into an address space of the client to render the service stub available for use to invoke the identified service; and

means for using the stub by the client to access the identified service.

20. A computer-readable medium containing instructions for controlling a data processing system to perform a method for accessing network services contained in a lookup service, the method comprising the steps of:

receiving a request by the lookup service for access to one of the network services; and

returning, by the lookup service, executable code to facilitate the access of the one network service.

21. The computer-readable medium of claim 20, wherein the method further includes the step of:

accessing the network service using the returned executable code.

22. The computer-readable medium of claim 20, wherein the returning step includes the step of:

returning a stub to the client.

23. The computer-readable medium of claim 20, wherein the returning step includes the step of:

returning a smart proxy to the client.

24. A computer-readable medium containing instructions for controlling a data processing system to perform a method for accessing network services contained in a lookup service, the method comprising the steps of:

sending to the lookup service a request to access one of the services; and

receiving executable code, from the lookup service and responsive to the request, that facilitates use of the one service.

25. The computer-readable medium of claim 24, wherein the method further includes the step of:

accessing the one service using the received executable code.

26. The computer-readable medium of claim 24, wherein the step of receiving executable code includes the step of:

receiving a stub that facilitates access of the one service.

27. The computer-readable medium of claim 24, wherein the step of receiving executable code includes the step of:

receiving a smart proxy that facilitates access of the one service.

* * * * *

EXHIBIT B

US005870719A

United States Patent [19]

Maritzen et al.

[11] Patent Number: 5,870,719

[45] Date of Patent: *Feb. 9, 1999

[54] **PLATFORM-INDEPENDENT, USAGE-INDEPENDENT, AND ACCESS-INDEPENDENT DISTRIBUTED QUOTE CONFIGURATON SYSTEM**

[75] Inventors: **L.M. Maritzen; Rolando D. Dimaandal**, both of Milpitas; **Julia Giannella**, Saratoga; **Raul Arregui**, Pacifica; **Marc Moss**, Fremont, all of Calif.

[73] Assignee: **Sun Microsystems, Inc.**, Mountain View, Calif.

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **675,683**

[22] Filed: **Jul. 3, 1996**

[51] **Int. Cl.**[6] **G06F 17/60**

[52] **U.S. Cl.** **705/26**; 345/329; 345/333; 345/334; 345/335; 364/400; 395/200.33; 395/200.47; 395/200.49; 395/200.6; 705/35; 707/104

[58] **Field of Search** 364/400, 464.1, 364/479.01, 479.02, 479.04, 479.06; 395/200.6, 200.33, 200.47, 200.48, 200.49, 201, 226, 227, 329, 330, 331, 333, 334, 335; 705/1, 26, 27, 400, 35; 345/329, 330, 331, 333, 334, 335; 707/104

[56] **References Cited**

U.S. PATENT DOCUMENTS

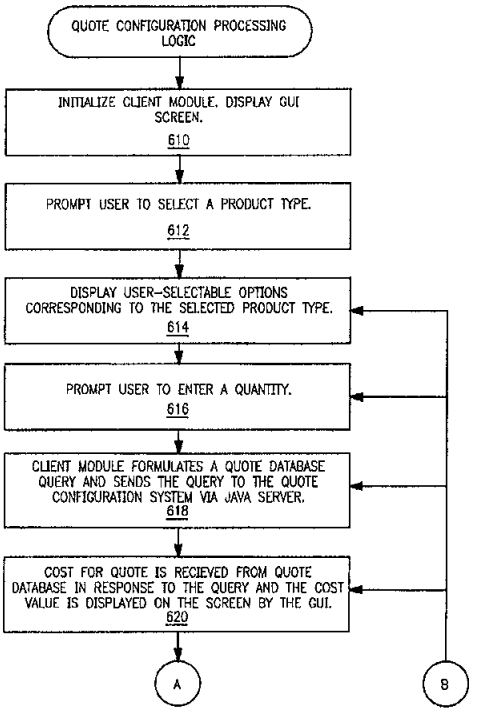
3,314,049	4/1967	Felcheck	705/15
4,473,824	9/1984	Claytor	340/825.27
5,331,543	7/1994	Yajima et al.	705/31
5,357,439	10/1994	Matsuzaki et al.	364/468.02
5,570,291	10/1996	Dudle et al.	364/468.01
5,666,493	9/1997	Wojcik et al.	705/26
5,694,546	12/1997	Reisman	395/200.9
5,710,887	1/1998	Chelliah et al.	395/226

Primary Examiner—Edward R. Cosimano
Attorney, Agent, or Firm—Blakely Sokoloff Taylor & Zafman

[57] **ABSTRACT**

A platform-independent, usage-independent, location-independent quote configuration system is described. The present invention, operating in a computer network, is a quote configurator comprising, 1) a client module, the client module having a platform-independent user interface for receiving quote input and command selections from a user, the quote input and command selections including product selection and selection of information indicative of business rules, and 2) a server coupled to the client module across the network, the server having access to quote data and business rules, the server including a platform-independent server interface configured to receive the quote input and command selections from the client module, the server validating the quote input based on the quote data and the business rules.

24 Claims, 7 Drawing Sheets



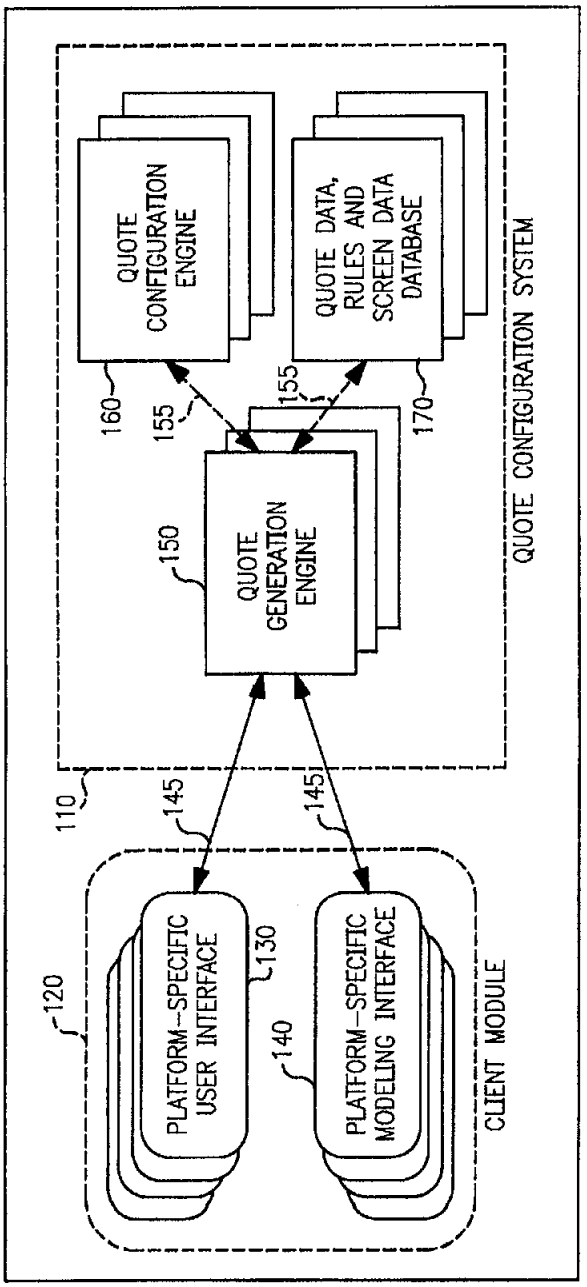


FIG. 1
PRIOR ART

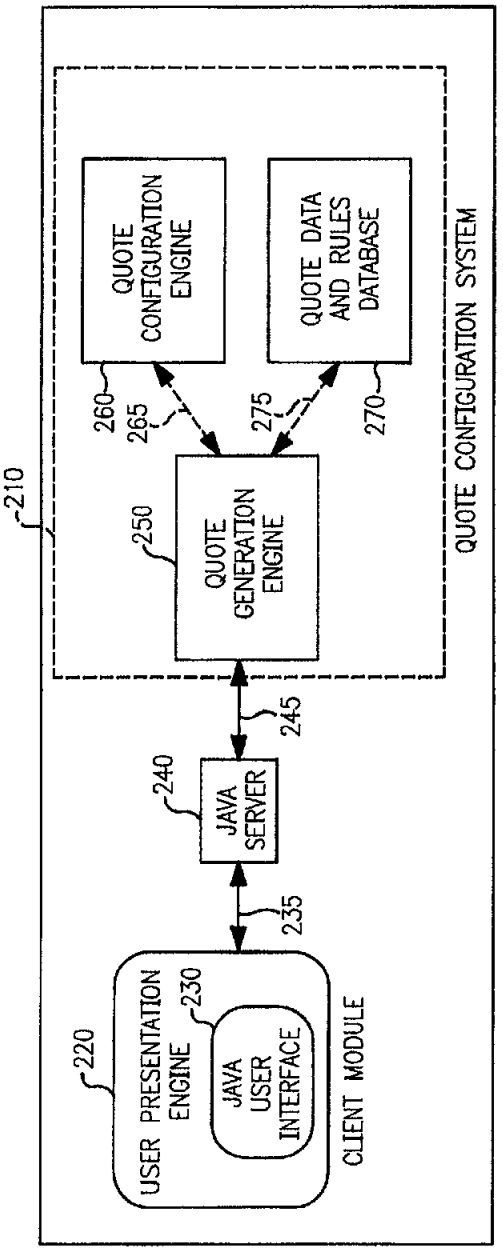

QUOTE CONFIGURATION SYSTEM
210
QUOTE CONFIGURATION ENGINE
260
265
QUOTE DATA AND RULES DATABASE
270
275
QUOTE GENERATION ENGINE
250
245
JAVA SERVER
240
235
USER PRESENTATION ENGINE
220
JAVA USER INTERFACE
230
CLIENT MODULE

FIG. 2

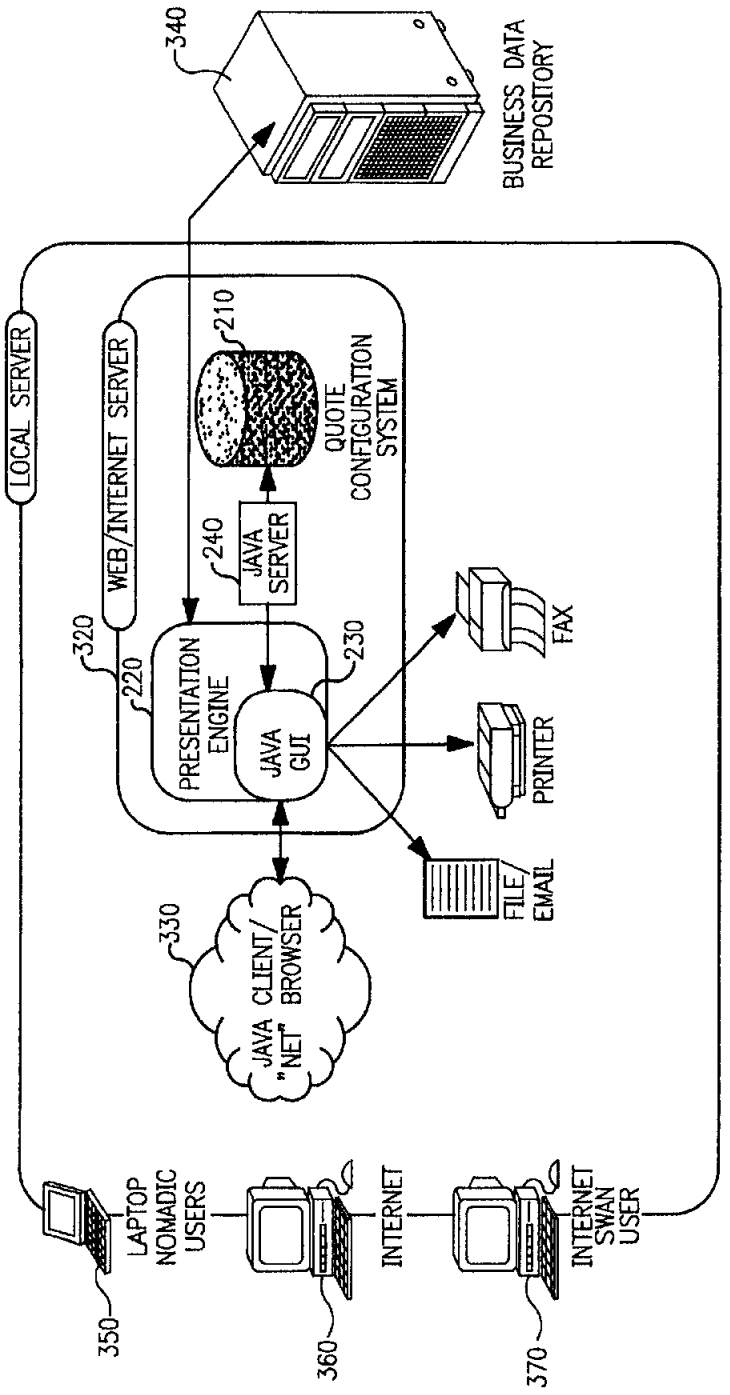

340
BUSINESS DATA REPOSITORY
LOCAL SERVER
WEB/INTERNET SERVER
210
QUOTE CONFIGURATION SYSTEM
240
JAVA SERVER
320
220
PRESENTATION ENGINE
230
JAVA GUI
FAX
PRINTER
FILE/ EMAIL
330
JAVA CLIENT/ "NET" BROWSER
LAPTOP NOMADIC USERS
INTERNET
INTERNET SWAN USER
350
360
370

FIG. 3

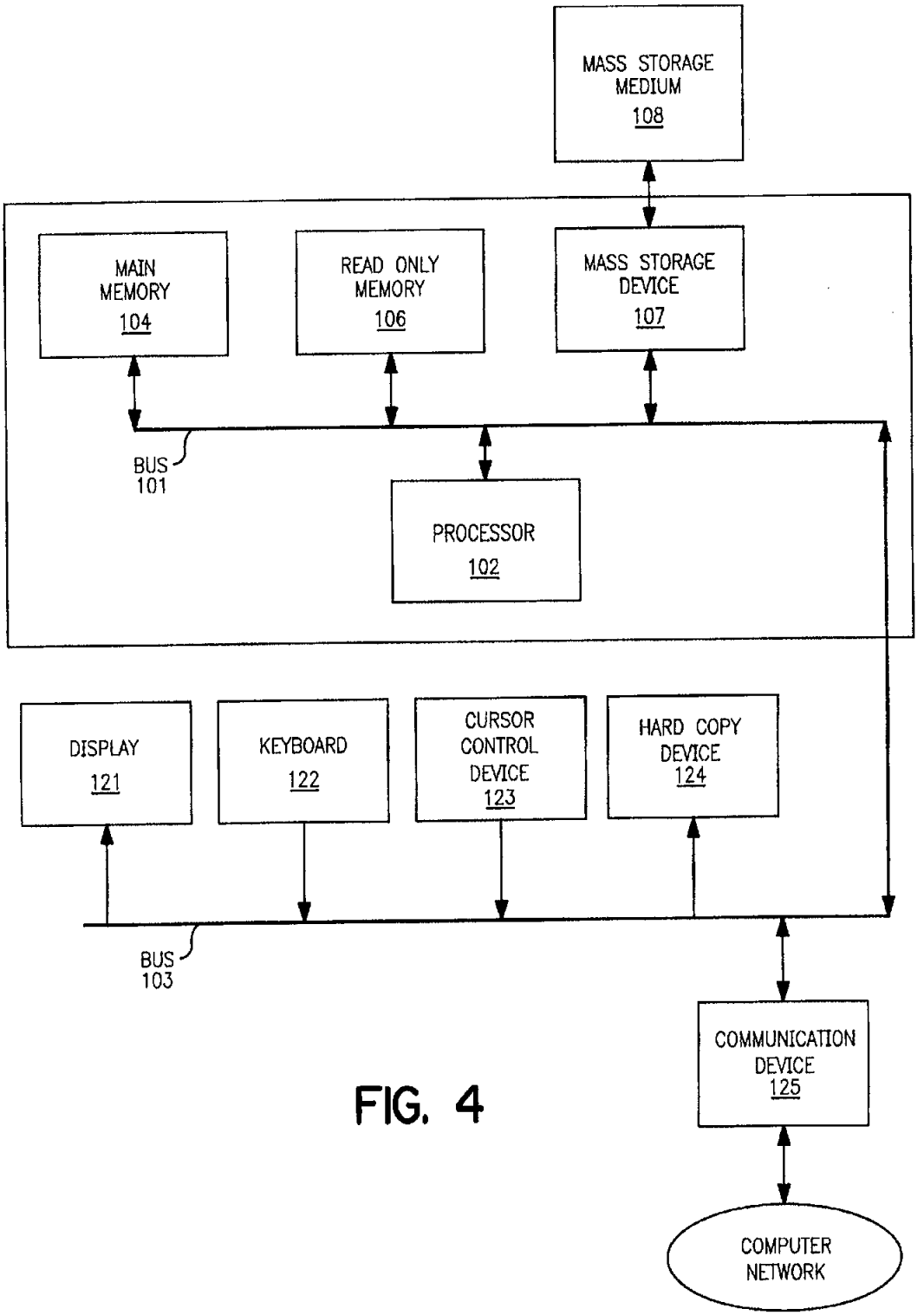

MASS STORAGE MEDIUM
108
MAIN MEMORY
104
READ ONLY MEMORY
106
MASS STORAGE DEVICE
107
BUS
101
PROCESSOR
102
DISPLAY
121
KEYBOARD
122
CURSOR CONTROL DEVICE
123
HARD COPY DEVICE
124
BUS
103
COMMUNICATION DEVICE
125
COMPUTER NETWORK

FIG. 4

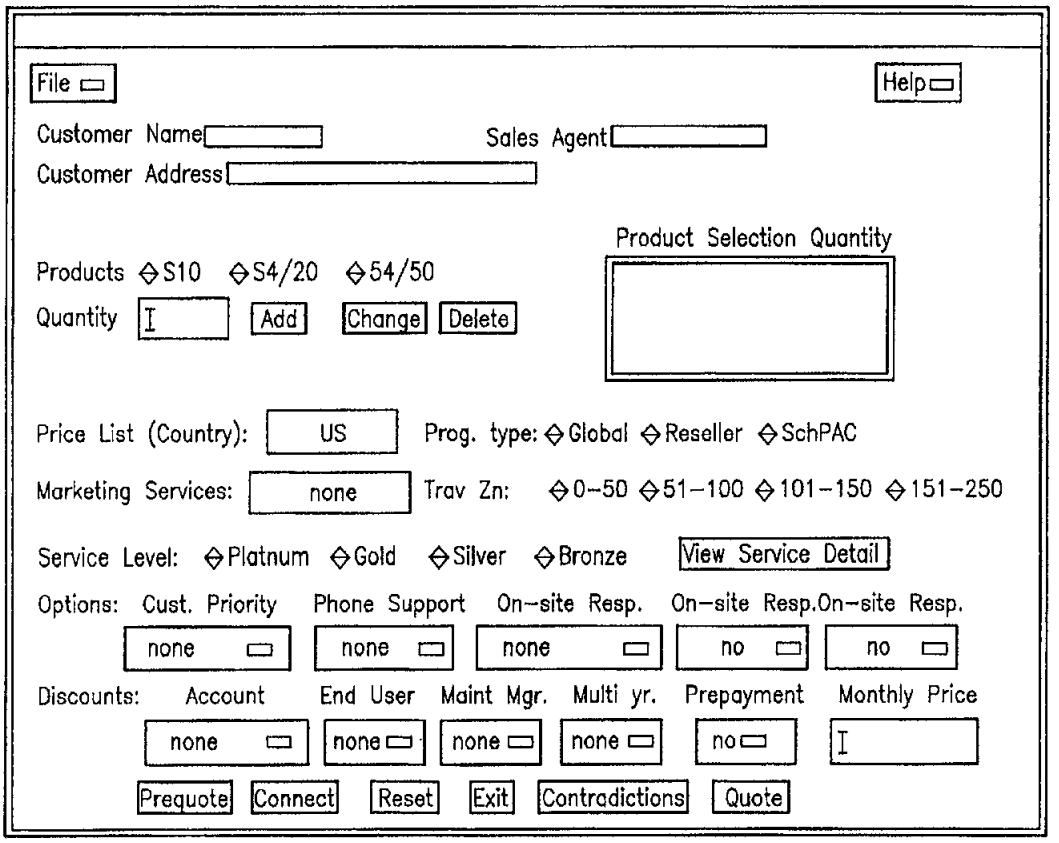

File
Help
Customer Name
Sales Agent
Customer Address
Product Selection Quantity
Products S10 S4/20 54/50
Quantity
Add
Change
Delete
Price List (Country):
US
Prog. type: Global Reseller SchPAC
Marketing Services:
none
Trav Zn: 0–50 51–100 101–150 151–250
Service Level: Platnum Gold Silver Bronze
View Service Detail
Options: Cust. Priority Phone Support On–site Resp. On–site Resp. On–site Resp.
none
none
none
no
no
Discounts: Account End User Maint Mgr. Multi yr. Prepayment Monthly Price
none
none
none
none
no
Prequote
Connect
Reset
Exit
Contradictions
Quote

FIG. 5

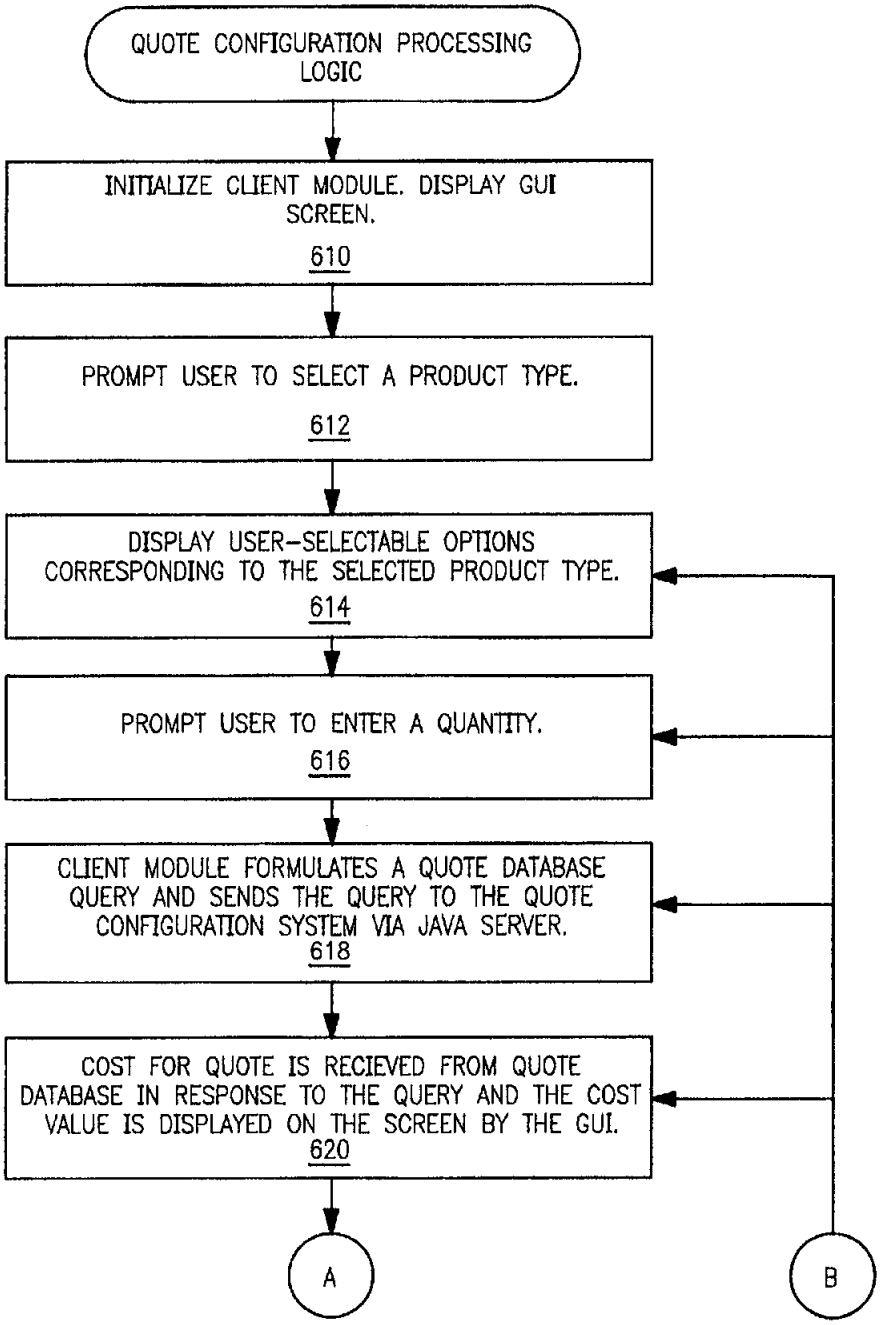

QUOTE CONFIGURATION PROCESSING LOGIC
INITIALIZE CLIENT MODULE. DISPLAY GUI SCREEN.
610
PROMPT USER TO SELECT A PRODUCT TYPE.
612
DISPLAY USER-SELECTABLE OPTIONS CORRESPONDING TO THE SELECTED PRODUCT TYPE.
614
PROMPT USER TO ENTER A QUANTITY.
616
CLIENT MODULE FORMULATES A QUOTE DATABASE QUERY AND SENDS THE QUERY TO THE QUOTE CONFIGURATION SYSTEM VIA JAVA SERVER.
618
COST FOR QUOTE IS RECIEVED FROM QUOTE DATABASE IN RESPONSE TO THE QUERY AND THE COST VALUE IS DISPLAYED ON THE SCREEN BY THE GUI.
620
A
B

FIG. 6

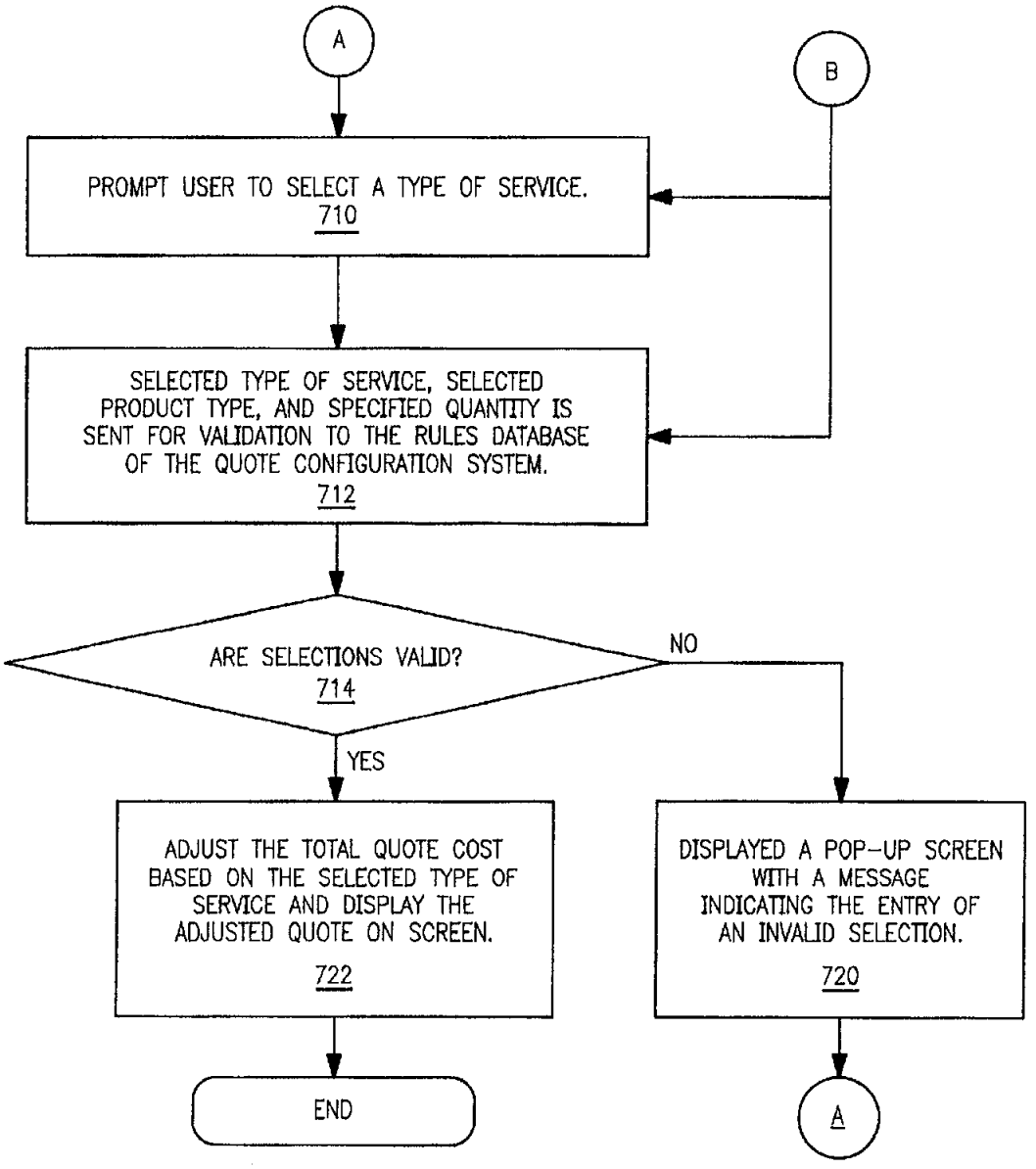

A
B
PROMPT USER TO SELECT A TYPE OF SERVICE.
710
SELECTED TYPE OF SERVICE, SELECTED PRODUCT TYPE, AND SPECIFIED QUANTITY IS SENT FOR VALIDATION TO THE RULES DATABASE OF THE QUOTE CONFIGURATION SYSTEM.
712
ARE SELECTIONS VALID?
714
NO
YES
ADJUST THE TOTAL QUOTE COST BASED ON THE SELECTED TYPE OF SERVICE AND DISPLAY THE ADJUSTED QUOTE ON SCREEN.
722
DISPLAYED A POP-UP SCREEN WITH A MESSAGE INDICATING THE ENTRY OF AN INVALID SELECTION.
720
END
A

FIG. 7

PLATFORM-INDEPENDENT, USAGE-INDEPENDENT, AND ACCESS-INDEPENDENT DISTRIBUTED QUOTE CONFIGURATON SYSTEM

FIELD OF THE INVENTION

The present invention relates to the field of distributed computer systems. Specifically, the present invention relates to the field of distributed computer systems for configuring and providing pricing and quotation information regardless of usage or access model and independent of platform type.

BACKGROUND OF THE INVENTION

Pricing and quotation information is produced at different points and different geographic locations in the sales process. Some of the key triggers used to produce this information include product sales, volume discounts, warranty terms, sales/lease contract terms, contract renewals or modifications, or special customer requests. Often times, quotations or quotes produced as a result of these triggers vary substantially based on the business unit or geographic location for which a particular quote is targeted. These variations are also caused by the different computer platforms, systems, usage models and networks used to assemble the quote information. Variations can also be caused by lack of, or differing, in-depth knowledge of the specific business rules and discounts that apply to each scenario, location, and business unit. In some cases, duplicate and redundant information must be reformatted or translated to a form compatible with a component of the quote generation system. In other cases, information must be manually entered. This is due in large part to the rigid data input requirements of conventional quote configuration systems.

The limitations in current quote configuration systems result in quotes that are difficult to generate and maintain, which contain errors that impact the customer and the business units, and require manual entry of redundant information at each step of the process.

Thus, a platform-independent, usage-independent, location-independent quote configuration systems which is capable of encapsulating the business logic and business unit expertise to produce consistent, accurate results is needed.

SUMMARY OF THE INVENTION

A platform-independent, usage-independent, location-independent quote configuration system is described. The present invention, operating in a computer network, is a quote configurator comprising, 1) a client module, the client module having a platform-independent user interface for receiving quote input and command selections from a user, the quote input and command selections including product selection and selection of information indicative of business rules, and 2) a server coupled to the client module across the network, the server having access to quote data and business rules, the server including a platform-independent server interface configured to receive the quote input and command selections from the client module, the server validating the quote input based on the quote data and the business rules.

Thus, it is an advantage of the present invention over conventional systems that the present invention is based on a non-procedural data-driven work flow design. It is a further advantage of the present invention that the point of service provides a real-time, interactive model for interaction on all components with the user. It is a further advantage of the present invention that all business rules and expertise needed to provide intelligent quoting are encapsulated within the quote configuration system, and are transferable and usable anywhere on any platform. This enables the design to be scalable with localization possible to accommodate variations in attributes such as currency and price list. It is a further advantage of the present invention that the present invention is based on an open architecture to enable integration with other systems, business processes and technologies. It is a further advantage of the present invention that the present invention is platform-independent (ex. SPARC™, RISC, X86). It is a further advantage of the present invention that the present invention is independent of usage model (ex. nomadic, remote/dial-up, inter-networked, intra-networked). It is a further advantage of the present invention that the present invention is independent of access model (ex. stand-alone application, networked application, web-based application, web-based applet).

The features and advantages of the present invention will be apparent from the accompanying drawings and from the detailed description of the preferred embodiment of the present invention as set forth below.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the accompanying drawings, in which:

FIG. **1** illustrates the system architecture of a prior art quote configuration system.

FIG. **2** illustrates the architecture of the quote configurator of the preferred embodiment of the present invention.

FIG. **3** illustrates the operation of the quote configurator of the preferred embodiment of the present invention.

FIG. **4** illustrates a typical data processing system or platform upon which one embodiment of the present invention is implemented.

FIG. **5** illustrates a sample screen display of the graphical user interface of the preferred embodiment.

FIGS. **6** and **7** are flowcharts illustrating the operation of the preferred embodiment.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The present invention is a platform-independent, usage-independent, location-independent quote configuration system. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one of ordinary skill in the art that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

FIG. **4** illustrates a typical data processing system or platform upon which one embodiment of the present invention is implemented. It will be apparent to those of ordinary skill in the art, however that other alternative systems of various system architectures may also be used. The data processing system illustrated in FIG. **4** includes a bus or other internal communication means **101** for communicating information, and a processor **102** coupled to the bus **101** for processing information. The system further comprises a random access memory (RAM) or other volatile storage device **104** (referred to as main memory), coupled to bus **101** for storing information and instructions to be executed by processor **102**. Main memory **104** also may be used for

storing temporary variables or other intermediate information during execution of instructions by processor **102**. The system also comprises a read only memory (ROM) and/or static storage device **106** coupled to bus **101** for storing static information and instructions for processor **102**, and a data mass storage device **107** such as a magnetic disk drive or optical disk drive. Data storage device **107** is coupled to bus **101** and is typically used with a computer readable mass storage medium **108**, such as a magnetic or optical disk, for storage of information and instructions. The system may further be coupled to a display device **121**, such as a cathode ray tube (CRT) or a liquid crystal display (LCD) coupled to bus **101** through bus **103** for displaying information to a computer user. An alphanumeric input device **122**, including alphanumeric and other keys, may also be coupled to bus **101** through bus **103** for communicating information and command selections to processor **102**. An additional user input device is cursor control **123**, such as a mouse, a trackball, stylus, or cursor direction keys coupled to bus **101** through bus **103** for communicating direction information and command selections to processor **102**, and for controlling cursor movement on display device **121**. Another device which may optionally be coupled to bus **101** through bus **103** is a hard copy device **124** which may be used for printing instructions, data, or other information on a medium such as paper, film, or similar types of media. In the preferred embodiment, a communication device **125** is coupled to bus **101** through bus **103** for use in accessing other nodes of a network computer system. This communication device **125** may include any of a number of commercially available networking peripheral devices such as those used for coupling to an Ethernet, token ring, Internet, or wide area network.

Note that any or all of the components of the system illustrated in FIG. **4** and associated hardware may be used in various embodiments of the present invention; however, it will be appreciated by those of ordinary skill in the art that any configuration of the system may be used for various purposes according to the particular implementation. In one embodiment of the present invention, the data processing system illustrated in FIG. **4** is an IBM® compatible personal computer (PC) or a SUN® SPARC™ Workstation. Processor **102** may be one of the 80X86 compatible microprocessors such as the 80486 or PENTIUM® brand microprocessors manufactured by INTEL® Corporation of Santa Clara, Calif.

The software implementing the present invention can be stored in main memory **104**, mass storage device **107**, or other storage medium locally accessible to processor **102**. It will be apparent to those of ordinary skill in the art that the methods and processes described herein can be implemented as software stored in main memory **104** or read only memory **106** and executed by processor **102**. This software may also be resident on an article of manufacture comprising a computer usable mass storage medium **108** having computer readable program code embodied therein and being readable by the mass storage device **107** and for causing the processor **102** to perform quote transactions and protocols in accordance with the teachings herein.

The preferred embodiment of the present invention is a platform-independent, usage-independent, location-independent quote configuration system, which can be implemented on computer system platforms such as the one illustrated in FIG. **4**.

ARCHITECTURE OF THE PREFERRED EMBODIMENT

FIG. **1** illustrates the typical quote configuration generation system currently used in the prior art. The architecture of a typical conventional quote processing system is based on a two-tier procedural model with deliberate data synchronization latency. The client module **120** is typically large, requiring significant local client resources (such as disk space, memory, swap space, etc.) to run the quote configuration system. The client module **120** includes a platform-specific modeling interface **140** and a platform-specific user interface **130**. The client module **120** is coupled to a conventional quote configuration system **110** across a network connection using a transaction messaging mechanism **145**, which is compatible with a conventional quote generation engine **150** in quote configuration system **110**. Quote generation engine **150** is then coupled to quote configuration engine **160** and quote database **170** across an interface using an intrasystem communication protocol **155**. Quote database **170** provides the quote data, rules, and screen data used by the quote generation engine **150** along with quote configuration engine **160** to generate a quote. Multiple configuration system databases are deployed to accommodate location-specific variables.

This prior art quote configuration methodology has several disadvantages, which are listed below:

1. A different platform-specific user interface **130** and client module **120** is necessary for each type of hardware architecture and operating system (OS) (i.e. platform) upon which the quote configuration system is deployed.

2. A different platform-specific user interface **130**, client module **120**, transaction messaging mechanism **145**, and intrasystem communication protocol **155** is necessary for each type of usage model for which the quote configuration system is implemented. In conventional systems, the four usage models commonly utilized are nomadic, remote/dial-up, inter-networked and intra-networked. In a nomadic system, the client module is resident in a portable computer system, which only periodically and asynchronously connects to the quote configuration system. In a remote/dial-up model, the client module uses a dial-up modem protocol to asynchronously connect with the quote configuration system **110**. In an inter-networked model, the client module uses Internet or World Wide Web (WWW) protocols to synchronously but indirectly connect to a quote configuration system. In an intra-networked model, the client module uses local area network (LAN) or wide-area network (WAN) protocols to synchronously and directly connect to a quote configuration system.

3. A different platform-specific user interface **130**, client module **120**, transaction messaging mechanism and intrasystem communication protocol is necessary for each type of access model for which the quote configuration system is implemented. The four access models commonly provided in conventional systems are stand-alone application, networked application, web-based application and web-based applet. Stand-alone applications are those software systems that run on a platform without network connectivity. Networked applications are those software systems that run on a platform with network connectivity and involve network transactions. The web-based models run on the World Wide Web (WWW) portion of the Internet and have dependencies on browser-specific implementations in addition to platform-specific dependencies. All four of these prior art access models have user interface screen rendering logic and data which is tightly integrated with the business rules and quote data in the quote configuration system **110**.

4. The system architecture and application programming interfaces (APIs) are proprietary and closed.

5. The work flow design is strictly procedural in a batch style of processing and has deliberate latency built into the data synchronization model.

6. Distribution of the quote configuration system and client software, synchronization of quote data and business rules, and version control of the system are difficult to manage effectively.

7. A different quote configuration engine **160** and quote database **170** must be deployed to accommodate each location-specific environment.

8. The user interface **130** requires extensive knowledge of the underlying business rules and quote data to generate a configuration quote.

9. The prior art design is based on a two-tiered, non-distributed/non-networked model.

FIG. **2** illustrates the quote configurator of the preferred embodiment of the present invention. In the present invention all common configurations and usage models are enabled by a single architecture and single user interface model, including mobile computing devices (nomadics and remote access/dial-ups), local area networks (LANs), wide area networks (WANs) and the Internet network. In general, these components provide specific functionality for handling quotation information and for generating quotes.

The architecture of the present invention is based on a distributed three-tier object-oriented design model as shown in FIG. 2. A first tier is a client module **220**. The second tier is a server **240** and the third tier is quote configuration system **210**.

The client module **220** includes a JAVA™ user interface **230**. JAVA is an object-oriented programming language similar to C++ developed by SUN® Microsystems of Mountain View, Calif. JAVA is unique in that a program written in JAVA is first compiled into byte-codes and subsequently interpreted into machine dependent processor instructions. Byte-codes are similar to machine instructions; however, byte-codes are not specific to a particular machine or platform. Thus, JAVA programs can run on any platform that supports JAVA. It is not necessary to recompile a JAVA program to run on a new machine. The JAVA user interface **230** may therefore be written without platform-specific code and without the need to create multiple versions that support multiple platforms.

Each tier of the system illustrated in FIG. **2** encapsulates all the necessary components for the intended function. All interfaces of the present invention are open and public. The messaging transport system of the present invention is based on socket-level communication protocol using open APIs at each end of the communications path, such as paths **235** and **245**. Socket-level communication protocol in the context of a computer network is well known to those of ordinary skill in the art. The user presentation engine of client module **220**, user interface **230**, and JAVA server **240** components are platform-independent, usage-independent, and access-independent and based on a single code source. The JAVA server **240** API which interfaces path **245** to quote generation engine **250** is an essential element of the present invention. Similarly, the JAVA server **240** API which interfaces path **235** to client module **220** is also an essential element of the present invention. Specifically, the user interface component **230** is a graphical user interface (GUI) and looks and operates in exactly the same manner independently of platform, usage, and access model. The GUI **230** is fully interactive and intelligent, and requires absolutely no knowledge of the underlying business rules and quote data for a user to generate an accurate configuration quote. The interaction of the GUI **230** is field-based, data-driven and event-driven. The quote data and rules database **270** also contains new components to support the present invention.

The client module **220** API to quote generation engine **250** through JAVA server **240** comprises commands, which are formulated in a platform-independent text string format and sent to quote generation engine **250** via a socket interface of JAVA server **240**. A specified port number is used by both the client module **220** and the quote generation engine **250**. The quote generation engine **250** interprets these commands and sends back its replies to the client module **220** via the socket interface of JAVA server **240**. A detailed description of the commands of this API follows.

Commands

bload rtssun7b.cab

This command makes the quote generation engine **250** read and load (in memory) all information contained in the rtssun7b.cab file. This file contains all business rules and screen rendering information needed by the quote generation engine **250**.

BEruntimeInit streamout

This command initiates runtime a communication protocol by the quote generation engine **250**. The 'streamout' argument is a parameter that tells the quote generation engine **250** to use its socket layer interface for communication.

BEinsPickByIndex <field> <idx> 1

This command instructs the quote generation engine **250** to pick a <field> with index <idx> and return all information stored for this field. The <field> argument can be any of the following in the preferred embodiment:

Contract	PriceBook	MarketingServices
ReplcHrdwrPrts	CstDefPrty	UnlimPhoneSprt
OnSiteResp	OnSiteResp2__hr	OnSiteCov7__24
PrefCustomer	EndUser	MaintMan
Multiyr		

The <indx> argument is an integer value representing the index of the specified field.

Example: BEinsPickByIndex Contract 2 1

(This sample command requests the quote generation engine **250** to retrieve all the information about the second field for contracts.)

BEinsValueSet NumberUsers val val <qty>

This command sets the quantity of products specified by the user. The <qty> argument is an integer value that represents the quantity of products.

BEreset

This command forces the quote generation engine **250** to ignore all previous input and start from the very beginning.

BEprequote

This command produces a pre-quote listing for the current session.

ueQuote

This command produces a final quote for the current session.

BEinsWhy <field> <state>

This command returns an explanation as to why a state of contradiction now exists for this <field>.

Example: BEinsWhy OnSiteResp2__hr 2 (This example asks why is there a contradiction in 2-hr response time?)

BEviewSummaryGet <windowname>

This command returns comprehensive information about a set of fields that logically belong to <windowname>.

Example: BEviewSummaryGet win4 (The quote generation engine **250** returns all current information and states of fields that belong to window **4**.)

These interfaces and commands of the preferred embodiment are only a subset of the entire API; however, these commands are sufficient for an operable system. It will be apparent to those of ordinary skill in the art that other commands or command arguments may also be provided. Conversely, the interfaces **265** and **275** to the quote configuration engine **260** and quote database **270**, respectively, are conventional interfaces.

OPERATION OF THE PREFERRED EMBODIMENT

FIG. **3** illustrates the operation of the quote configurator of the preferred embodiment of the present invention. The client module **220** communicates with JAVA server **240** and quote configuration system **210** in the context of the World Wide Web (WWW) and the Internet **320**. The client module **220** may also gather other business data from repository **340**. Network users of the present invention such as nomadic users **350**, Internet users **360**, or intranet users **370** may access the client module **220** of the present invention via JAVA client/net browser **330**. Once these network users connect to the client module **220**, the users may request and view quote information. Quote information may also be printed, faxed, filed, or emailed via output devices connected to client module **220**.

The following sample scenario will demonstrate an example of the quote configuration process of the present invention for generating a custom quote from both the user and system perspective. This example will highlight the essential steps of the present invention as appropriate. In this description, no distinction is made regarding location of the user/system, access method of the user/system, or the type of platform on which either the user or system exists. The same process and model will work regardless of platform, access type, or usage model. FIGS. **2** and **3** are referenced in combination with the flow diagrams of FIGS. **6** and **7** to illustrate the operation of the present invention. In addition, the screen sample shown in FIG. **5** also referenced.

On initiation of the client module **220** of the present invention by the user, the GUI **230** establishes a socket connection with the JAVA server **240** in real-time, and loads an initial data set. In block **610**, shown in FIG. **6**, a main screen or window of the GUI **230** is then auto-populated with a series of default values as specified by the JAVA GUI **230**, based on the encapsulated business rules. One such sample screen is illustrated in FIG. **5**. These default values include Price Lists; Travel Zone; Service Level. At this point in the process, the user only needs to select a product type and enter a quantity in order to receive a quote. In this example, the user will be producing a custom quote.

1. User selects the product.

The user is prompted in block **612**, shown in FIG. **6**, with a pick list to select a product type or the user can ask for additional information/help. On selection of the product type from the pick list presented by the GUI **230**, the GUI **230** logic updates the screen with appropriate user-selectable options, key hints and other information intended to assist the user with creation of the custom quote based on the user selection of product type (block **614**).

2. User enters a quantity.

On entry of the quantity (block **616**), the GUI **230** logic determines the initial total cost for the quote by using the JAVA server **240** and socket protocol via path **245** to query the quote database **270** (block **618**), perform the calculations and update the appropriate portion of the screen of the GUI **230** (block **620**). As denoted by bubble B shown in FIGS. **6** and **7**, the user may iteratively select or edit the selections described in blocks **614–712**.

3. User selects the types of coverage desired.

Referring now to FIG. **7**, selection of the type of coverage provides an adjustment in total quote cost based on the type service selected (blocks **710** and **712** shown in FIG. **7**). This selection of type of coverage identifies a set of business rules which is applied to the request. If this type of coverage is not available for the selected product type and the specified quantity, the business rule of the quote configuration system quote database will indicate an invalid condition which is communicated to the GUI **230** via JAVA server **240** (block **714**). This invalid condition causes a pop-up screen to appear for the user (block **720**). The pop-up screen contains information and navigation/event assistance for the user. These pop-up screens indicating invalid conditions, warnings and hints can be displayed at any time and triggered by any appropriate event; because, the JAVA server **240** and quote configuration system **210** are in frequent communication regarding of the state of the GUI **230** and selected events/options. Also, the user can "override" the suggestion/warning pop-up screen message and proceed with the quote anyway in order to accommodate local variations/conditions.

In addition to event-driven assistance, the user can get on-demand assistance on invalid conditions, additional hints and warnings by selecting the "Contradictions" button as shown on the screen sample shown in FIG. **5**. After the quote is adjusted based on the selected product type, the specified quantity, and type of service, the adjusted quote is displayed for the user on the screen (block **722**). The user can view and process the quote at any time during the configuration process.

The user has the following additional options provided at any time during the quote process in the preferred embodiment: 1) view the quote on-line; 2) fax the quote to a fax-enabled destination using a conventional facsimile device; 3) e-mail the quote to a networked destination using a conventional electronic mail system; 4) print the quote to a system-acknowledged printer; or 5) save the quote to a file. Buttons for selecting these options are shown in FIG. **5** and the interface supporting these options is shown in FIG. **3**.

Thus, a platform-independent, usage-independent, and access-independent quote configuration system has been described. The specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation to a particular embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the following claims.

We claim:

1. In a computer network, a quote configurator comprising:

a client module, the client module having a user interface to receive quote input and command selections from a user, the quote input and command selections including product selection and selection of information indicative of business rules; and

a server coupled to the client module across the network, the server having access to quote data and business rules, the server including a server interface configured to receive the quote input and command selections from the client module, the server validating the quote input based on the quote data and the business rules,

said user interface and said server interface being comprised of platform independent byte-codes.

2. The quote configurator as claimed in claim 1 wherein the server interface is a platform-independent, usage-independent and access-independent transport protocol for messaging quote data and business rules between the server and the client module.

3. The quote configurator as claimed in claim 1 wherein the user interface is a platform-independent, usage-independent and access-independent graphical user interface.

4. The quote configurator as claimed in claim 1 wherein the client module is JAVA compatible.

5. The quote configurator as claimed in claim 1 wherein the server is JAVA compatible.

6. The quote configurator as claimed in claim 1 wherein the client interface byte-codes are interpreted for operation on a particular client platform.

7. The quote configurator as claimed in claim 1 wherein the server interface byte-codes are interpreted for operation on a particular server platform.

8. The quote configurator as claimed in claim 1 wherein the user interface includes a screen display configured for product selection and selection of information indicative of business rules.

9. The quote configurator as claimed in claim 1 wherein the client module further includes an interface to a nomadic user.

10. The quote configurator as claimed in claim 1 wherein the client module further includes an interface to an Internet user.

11. The quote configurator as claimed in claim 1 wherein the client module further includes an interface to an intranet user.

12. A method for configuring quotes comprising:

receiving quote input and command selections from a user via a user interface in a client module, the quote input and command selections including product selection and selection of information indicative of business rules;

sending said quote input and command selections to a server across a server interface, said user interface and server interface being comprised of platform independent byte-codes;

accessing quote data and business rules; and

validating the quote input based on the quote data and the business rules.

13. The method as claimed in claim 12 wherein the sending further includes preparing the quote input and command selections in a platform-independent, usage-independent and access-independent transport protocol for transport to the server.

14. The method as claimed in claim 12 wherein the user interface is a platform-independent, usage-independent and access-independent graphical user interface.

15. The method as claimed in claim 12 wherein the client module is JAVA compatible.

16. The method as claimed in claim 12 wherein the server is JAVA compatible.

17. The method as claimed in claim 12 further including interpreting the client interface byte-codes for operation on a particular client platform.

18. The method as claimed in claim 12 further including interpreting the server interface byte-codes for operation on a particular server platform.

19. The method as claimed in claim 12 further including receiving a product selection and receiving a selection of information indicative of business rules.

20. The method as claimed in claim 12 further including receiving input from a nomadic user.

21. The method as claimed in claim 12 further including receiving input from an Internet user.

22. The method as claimed in claim 12 further including receiving input from an intranet user.

23. On a computer useable medium having computer readable code embodied therein, a client module for causing the computer to configure quotes, the client module comprising:

computer readable program code configured to cause the computer to receive quote input and command selections from a user via a user interface, the quote input and command selections including product selection and selection of information indicative of business rules;

computer readable program code configured to cause the computer to send said quote input and command selections to a remote server across a server interface; and

computer readable program code configured to cause the computer to receive validation information from the remote server indicating whether the quote input is valid based on quote data and the business rules,

said user interface and said server interface being comprised of platform independent byte-codes.

24. On a computer useable medium having computer readable code embodied therein, a server for causing the computer to configure quotes, the server comprising:

computer readable program code configured to cause the computer to receive quote input and command selections from a remote client module across a server interface;

computer readable program code configured to cause the computer to access quote data and business rules; and

computer readable program code configured to cause the computer to validate the quote input based on the quote data and the business rules,

said server interface being comprised of platform independent byte-codes.

* * * * *

EXHIBIT C

US005963950A

United States Patent [19]

Nielsen et al.

[11] Patent Number: 5,963,950

[45] Date of Patent: Oct. 5, 1999

[54] **METHOD AND SYSTEM FOR FACILITATING ACCESS TO SELECTABLE ELEMENTS ON A GRAPHICAL USER INTERFACE**

[75] Inventors: **Jakob Nielsen**, Atherton; **Earl Johnson**, San Jose; **Donald R. Gentner**, Palo Alto, all of Calif.

[73] Assignee: **Sun Microsystems, Inc.**, Palo Alto, Calif.

[21] Appl. No.: **08/670,952**

[22] Filed: **Jun. 26, 1996**

[51] **Int. Cl.**[6] **G06F 17/30**

[52] **U.S. Cl.** **707/102**; 707/2; 707/500; 707/104

[58] **Field of Search** 707/500, 104, 707/102, 2; 345/333; 395/200.47

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,263,174	11/1993	Layman	345/800
5,297,249	3/1994	Berstein	345/356
5,339,392	8/1994	Risberg	345/338
5,572,643	11/1996	Judson	395/200.48
5,603,025	2/1997	Tabb	707/2
5,621,871	4/1997	Jaremko	345/441
5,659,729	8/1997	Nielsen	707/3
5,692,073	11/1997	Cass	382/219
5,708,825	1/1998	Sotomayor	1/1
5,787,416	7/1998	Tabb	707/2
5,806,077	9/1998	Wecker	707/501
5,877,766	3/1999	Bates	345/357

FOREIGN PATENT DOCUMENTS

0 328 831 A2	8/1989	European Pat. Off. .
0 428 261 A2	5/1991	European Pat. Off. .
94/17469	8/1994	WIPO .

OTHER PUBLICATIONS

"Multi–Character H–To–Hawaii Search Capability", IBM Technical Disclosure Bulletin, vol. 37, No. 12, Dec. 1994, pp. 285–286, XP000487787.

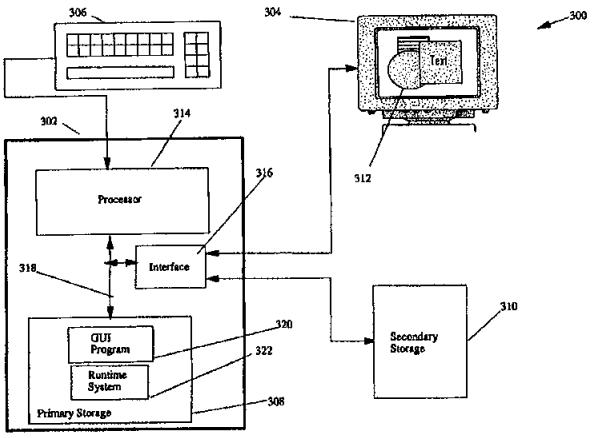
Jon Gunderson, Ph.D., "World Wide Web Browser Access Recommendations", University of Illinois at Urbana/Champaign, no date.

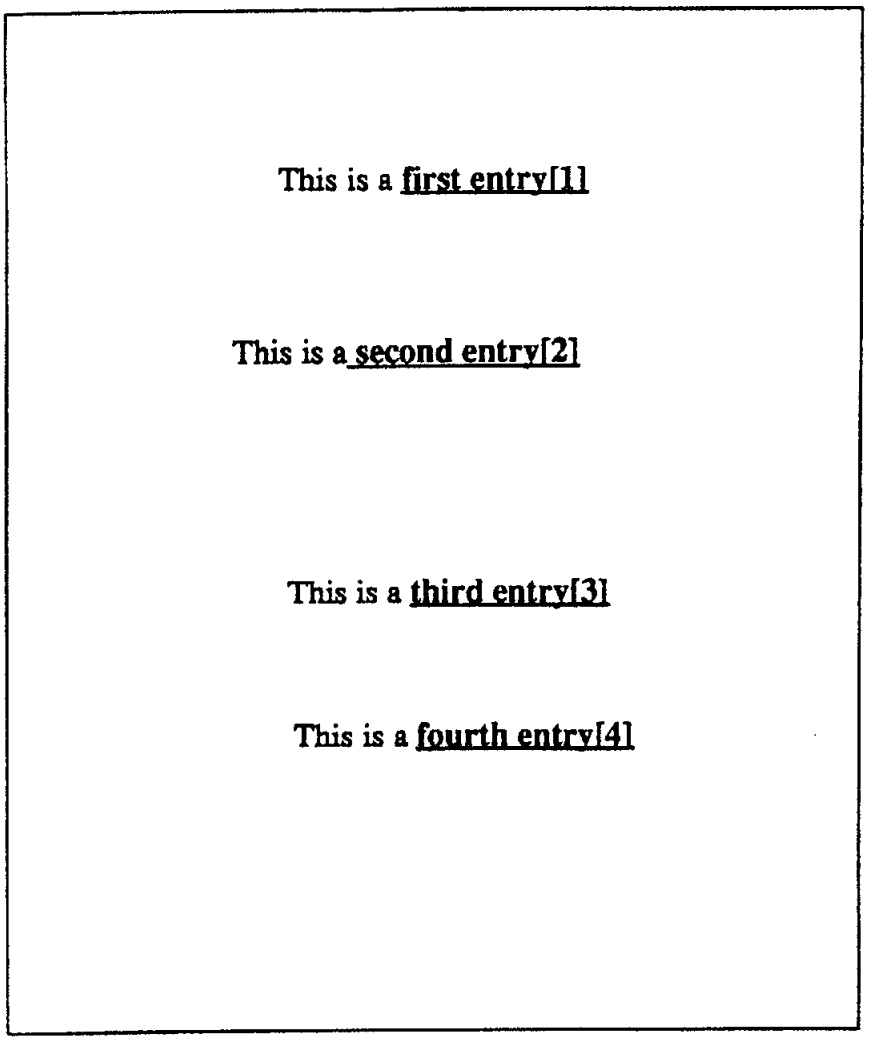
Primary Examiner—Thomas G. Black
Assistant Examiner—Diane D. Mizrahi
Attorney, Agent, or Firm—McDermott, Will & Emery

[57] **ABSTRACT**

In one embodiment the present invention provides a method of selecting a selectable element with a character input device from a list of exically unordered selectable elements on a graphical user interface (GUI). Typically, these selectable elements include hypertext links and GUI buttons on the GUI of computer program executing on a computer system. Each selectable element includes a character portion of data which facilitates selecting the selectable element. The method typically begins the selection process when a user enters one or more characters from a character input device. In response to receiving the characters, the present invention inserts each character into a match string. The match string is then compared with the character portion of each selectable element on the GUI. A selectable element is "armed" when the character portion of a selectable element is found which matches the match string. Arming a selectable element prepares the element for selection and typically involves highlighting, underlining or otherwise distinguishing the hypertext link from other text on a page. The user can then select the armed selectable element by actuating the "enter" key. If a hypertext selectable element is selected then the information associated with a destination node (e.g., a web page) is displayed for the user. Instead of selecting the armed selectable element, the user may search for a different selectable element by entering additional characters and not actuating the "enter" key. The method disclosed provides an improved selection method over prior methods because a user can quickly search and select a selectable element by typing a minimum number of characters.

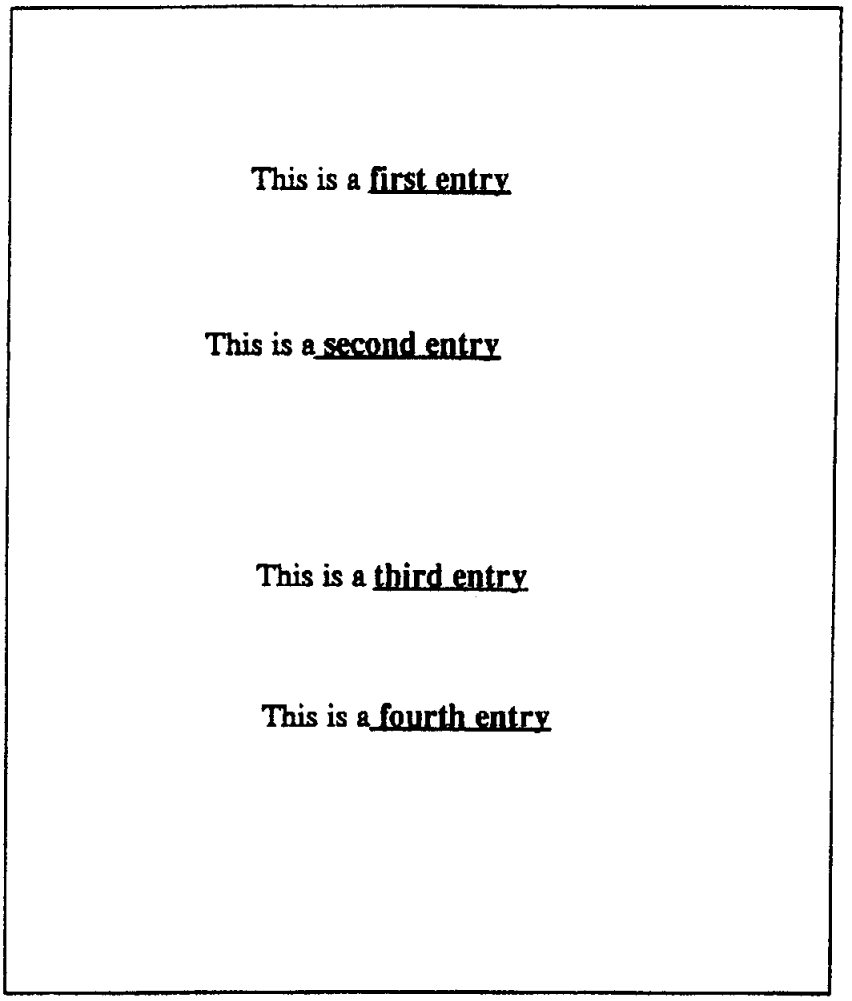
26 Claims, 11 Drawing Sheets





Prior Art CERN Interface

Figure 1



Prior Art LYNX Interface

Figure 2

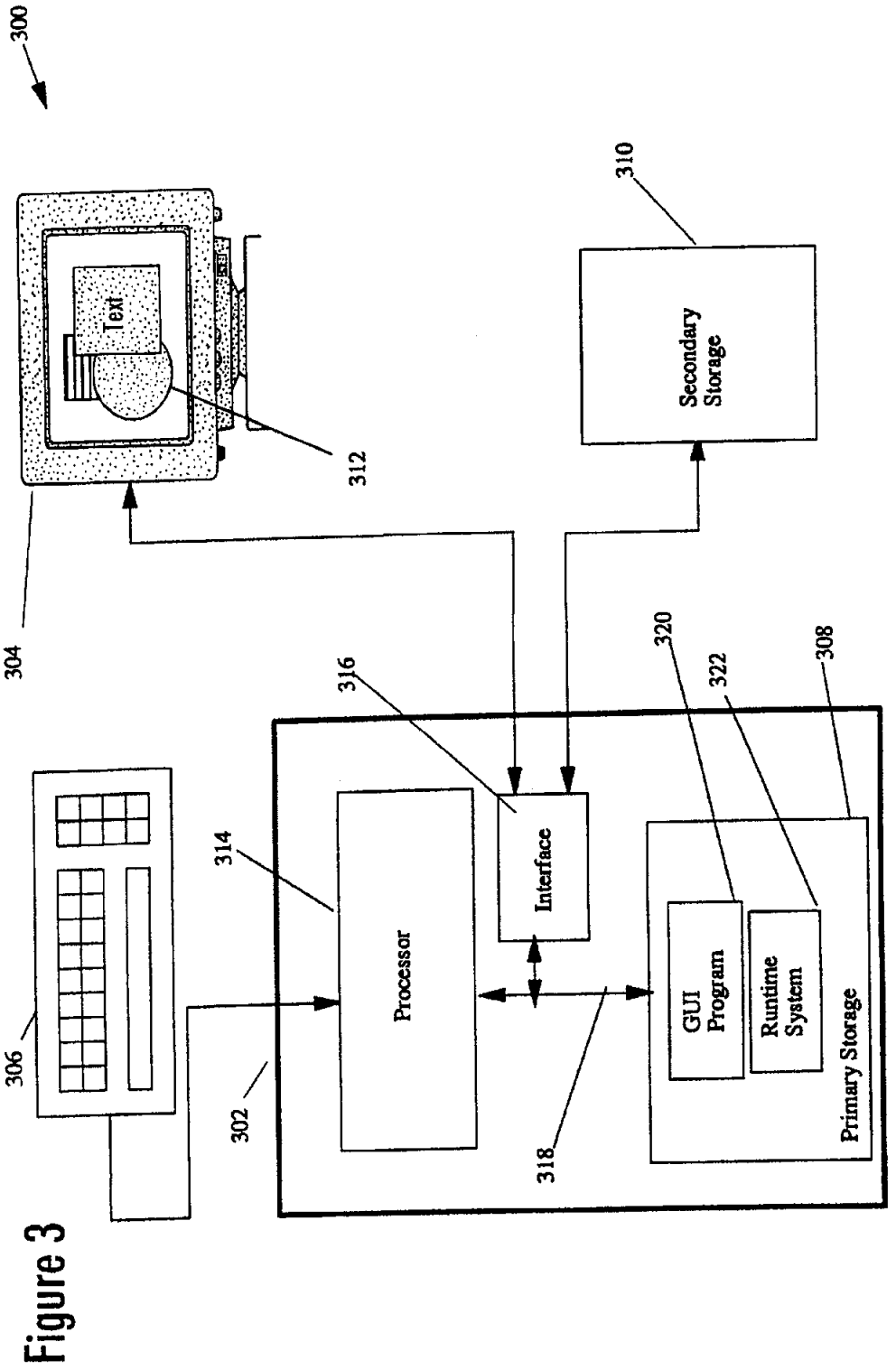

300
310
Secondary Storage
Text
312
304
316
320
322
308
314
Interface
Processor
GUI Program
Runtime System
Primary Storage
306
302
318

Figure 3

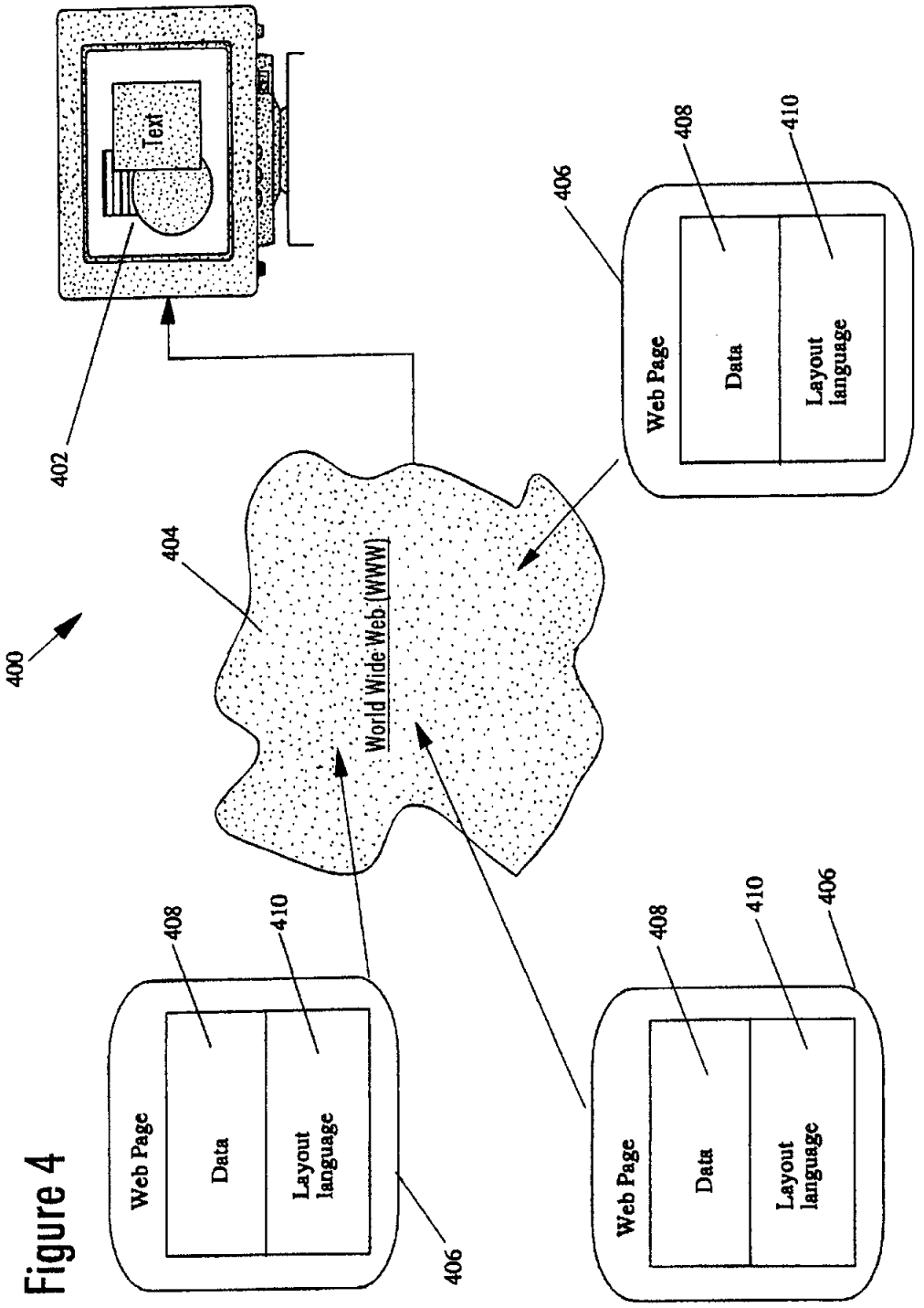

Text
402
404
World Wide Web (WWW)
400
406
408
410
Web Page
Data
Layout language

Figure 4

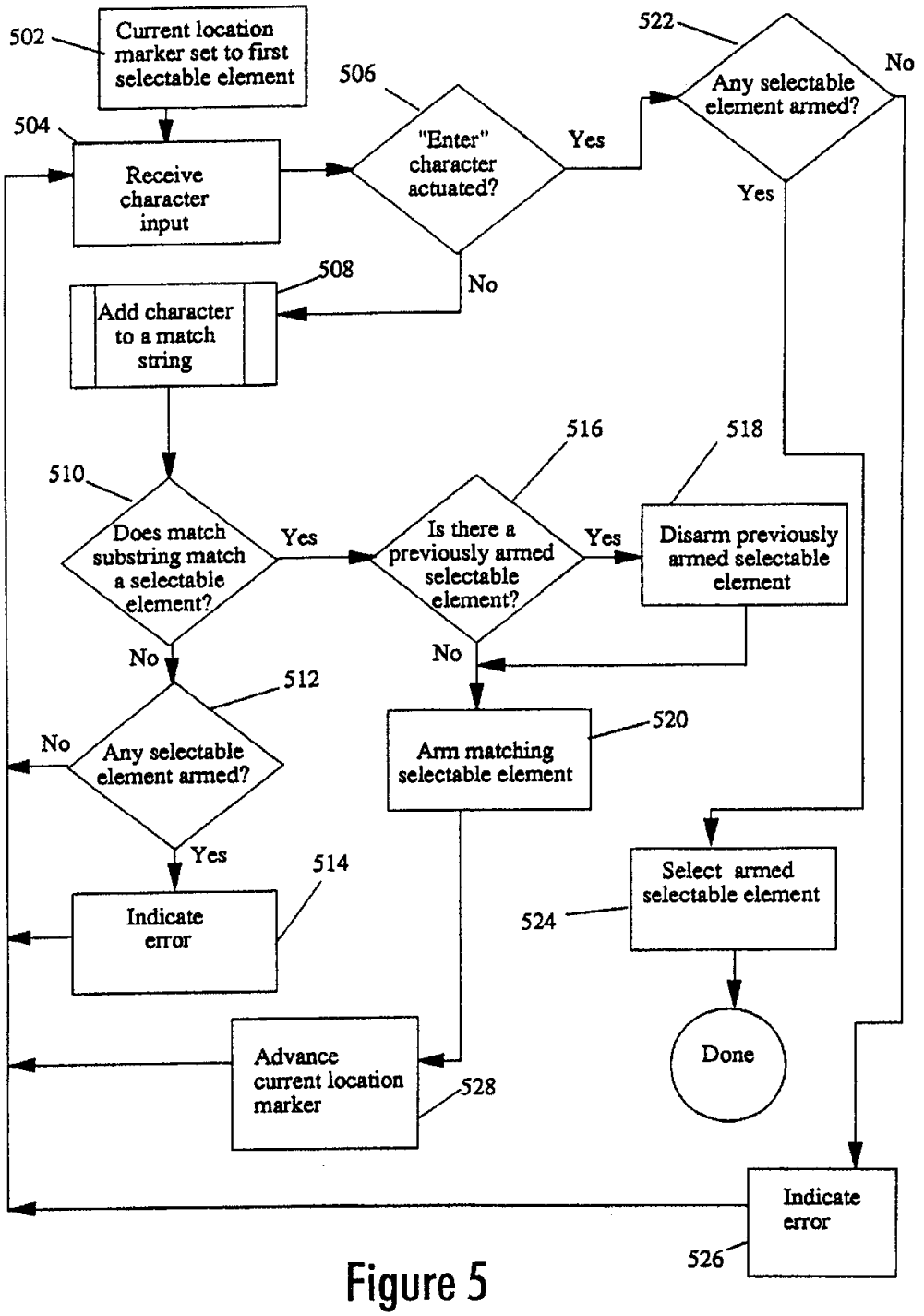

502
Current location marker set to first selectable element
504
Receive character input
506
"Enter" character actuated?
Yes
No
522
Any selectable element armed?
No
Yes
508
Add character to a match string
510
Does match substring match a selectable element?
Yes
No
516
Is there a previously armed selectable element?
Yes
No
518
Disarm previously armed selectable element
512
Any selectable element armed?
No
Yes
520
Arm matching selectable element
514
Indicate error
524
Select armed selectable element
Advance current location marker
528
Done
Indicate error
526

Figure 5

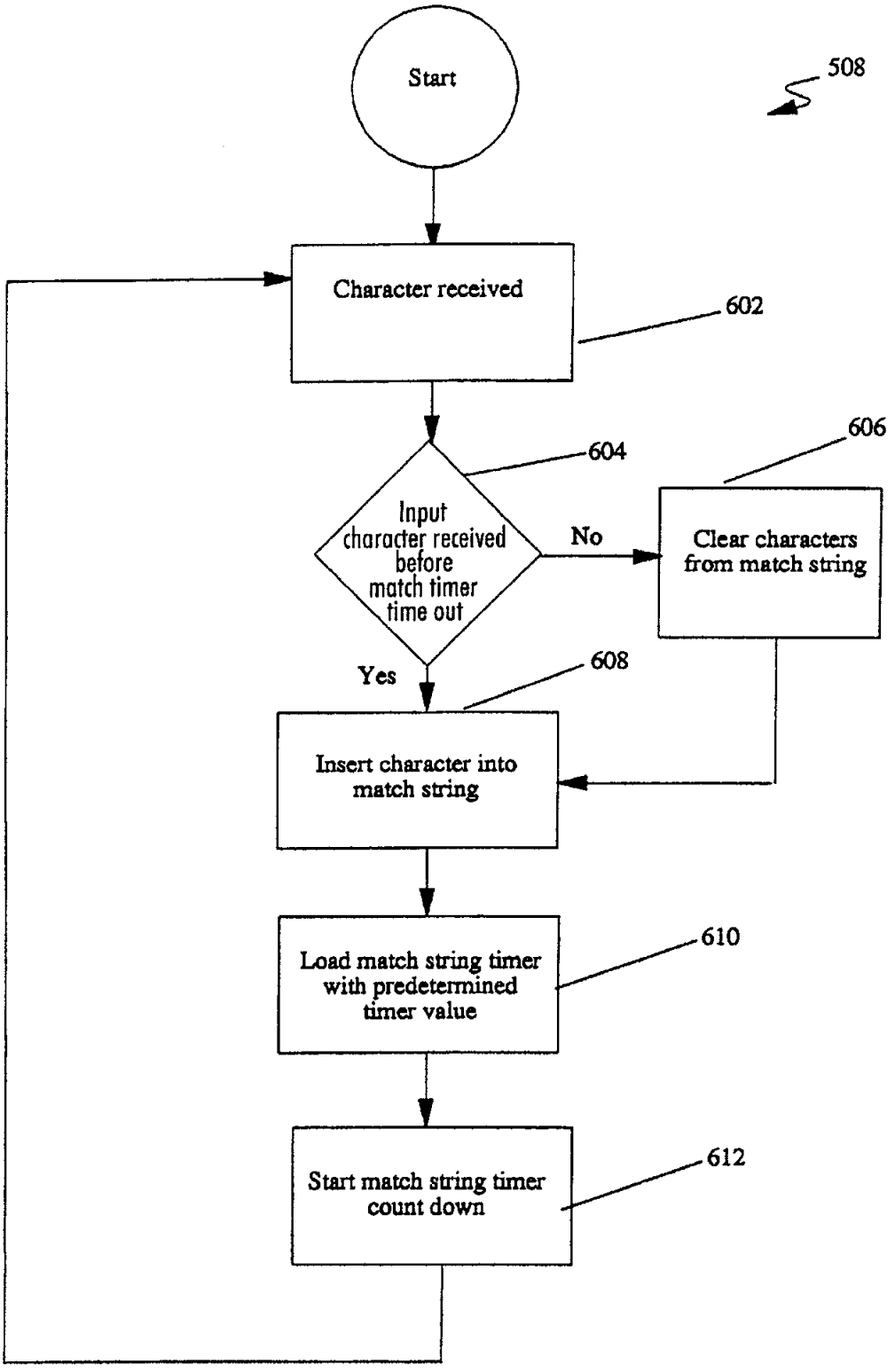

Start
508
Character received
602
604
606
Input character received before match timer time out
No
Clear characters from match string
Yes
608
Insert character into match string
Load match string timer with predetermined timer value
610
Start match string timer count down
612

Figure 6

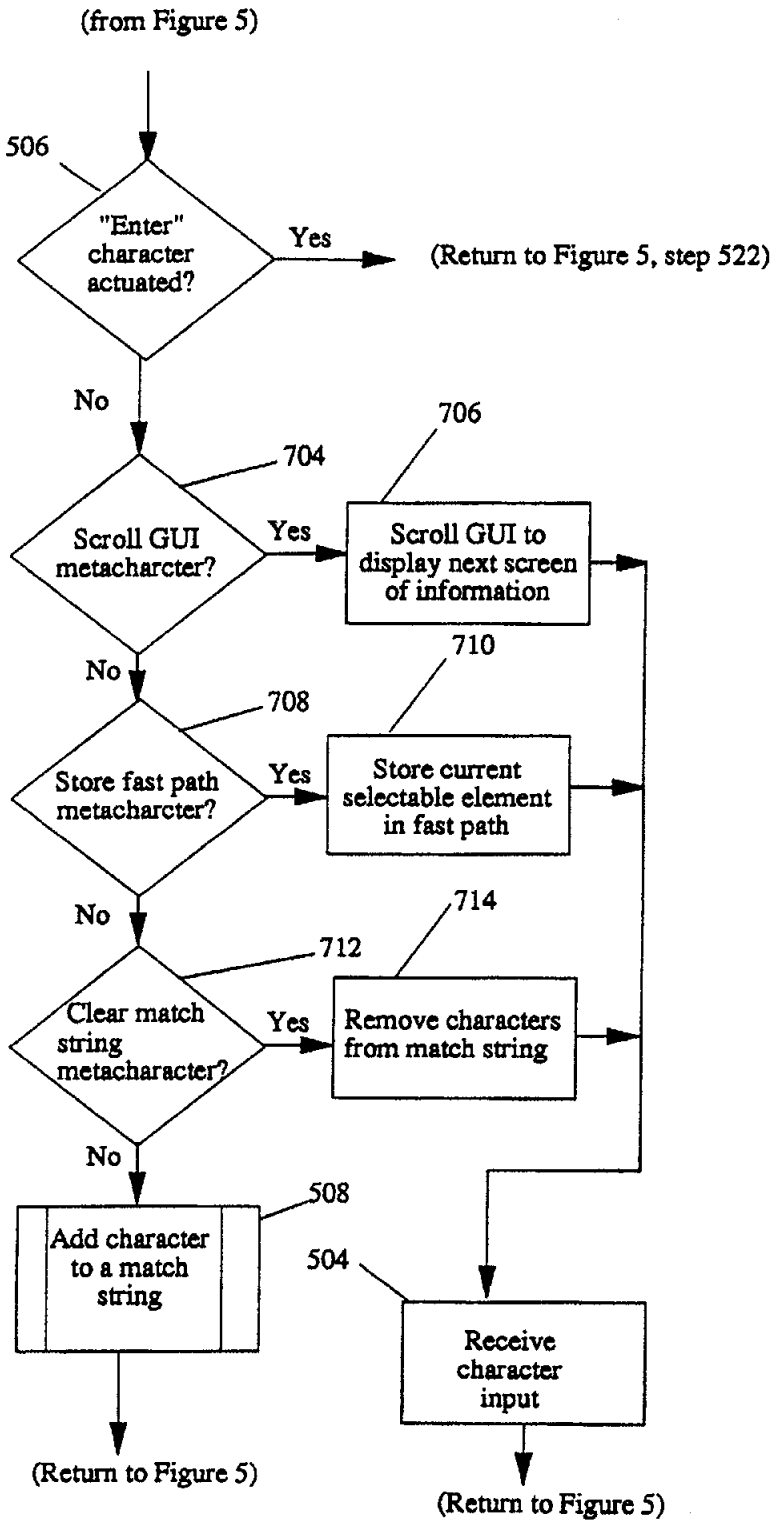

(from Figure 5)
506
"Enter" character actuated?
Yes
(Return to Figure 5, step 522)
No
704
Scroll GUI metacharcter?
Yes
706
Scroll GUI to display next screen of information
No
708
Store fast path metacharcter?
Yes
710
Store current selectable element in fast path
No
712
Clear match string metacharacter?
Yes
714
Remove characters from match string
No
508
Add character to a match string
504
Receive character input
(Return to Figure 5)
(Return to Figure 5)

Figure 7

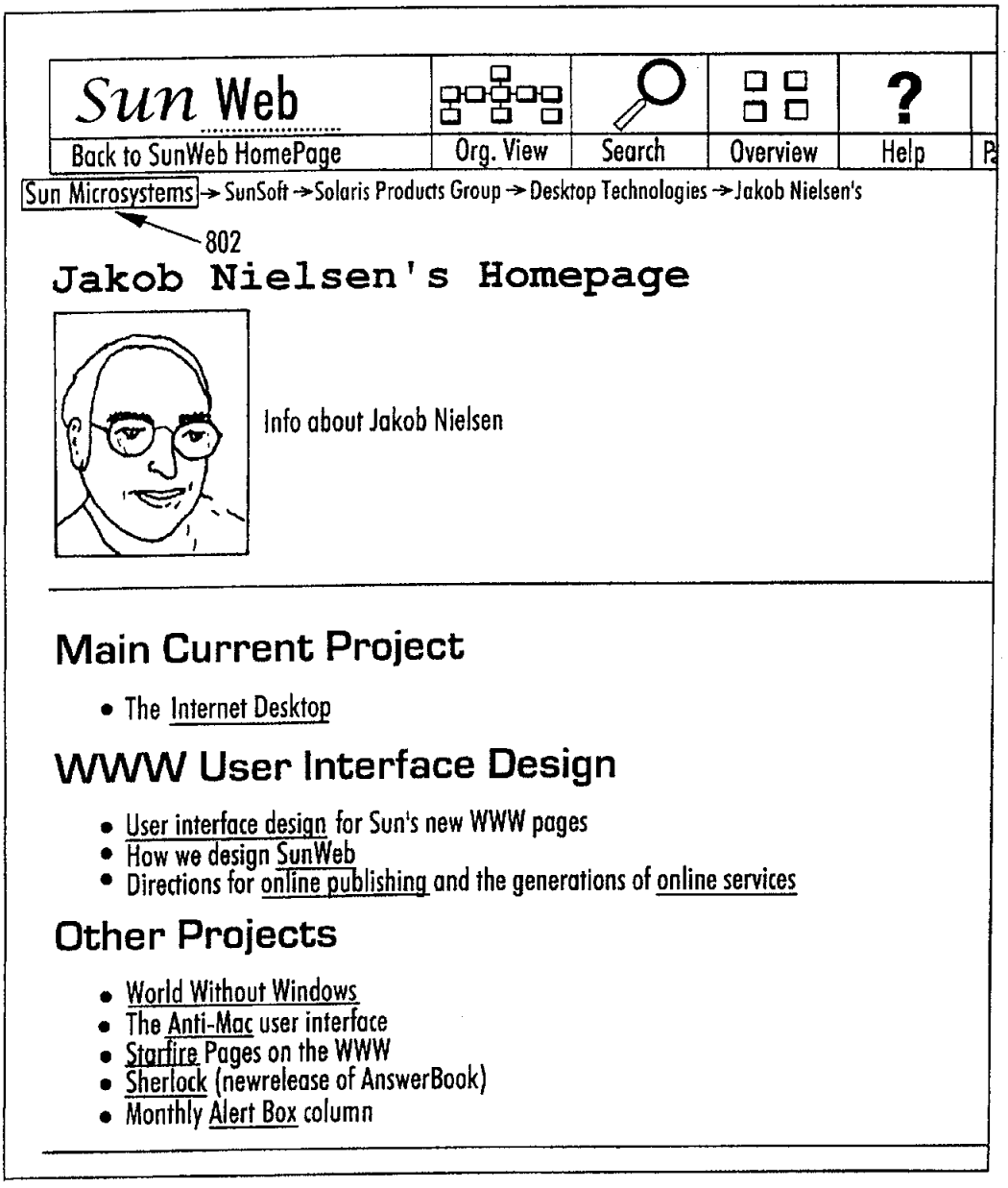

Sun Web
Back to SunWeb HomePage
Org. View
Search
Overview
Help
Sun Microsystems → SunSoft → Solaris Products Group → Desktop Technologies → Jakob Nielsen's
802
Jakob Nielsen's Homepage
Info about Jakob Nielsen
Main Current Project
The Internet Desktop
WWW User Interface Design
User interface design for Sun's new WWW pages
How we design SunWeb
Directions for online publishing and the generations of online services
Other Projects
World Without Windows
The Anti-Mac user interface
Starfire Pages on the WWW
Sherlock (newrelease of AnswerBook)
Monthly Alert Box column

Figure 8

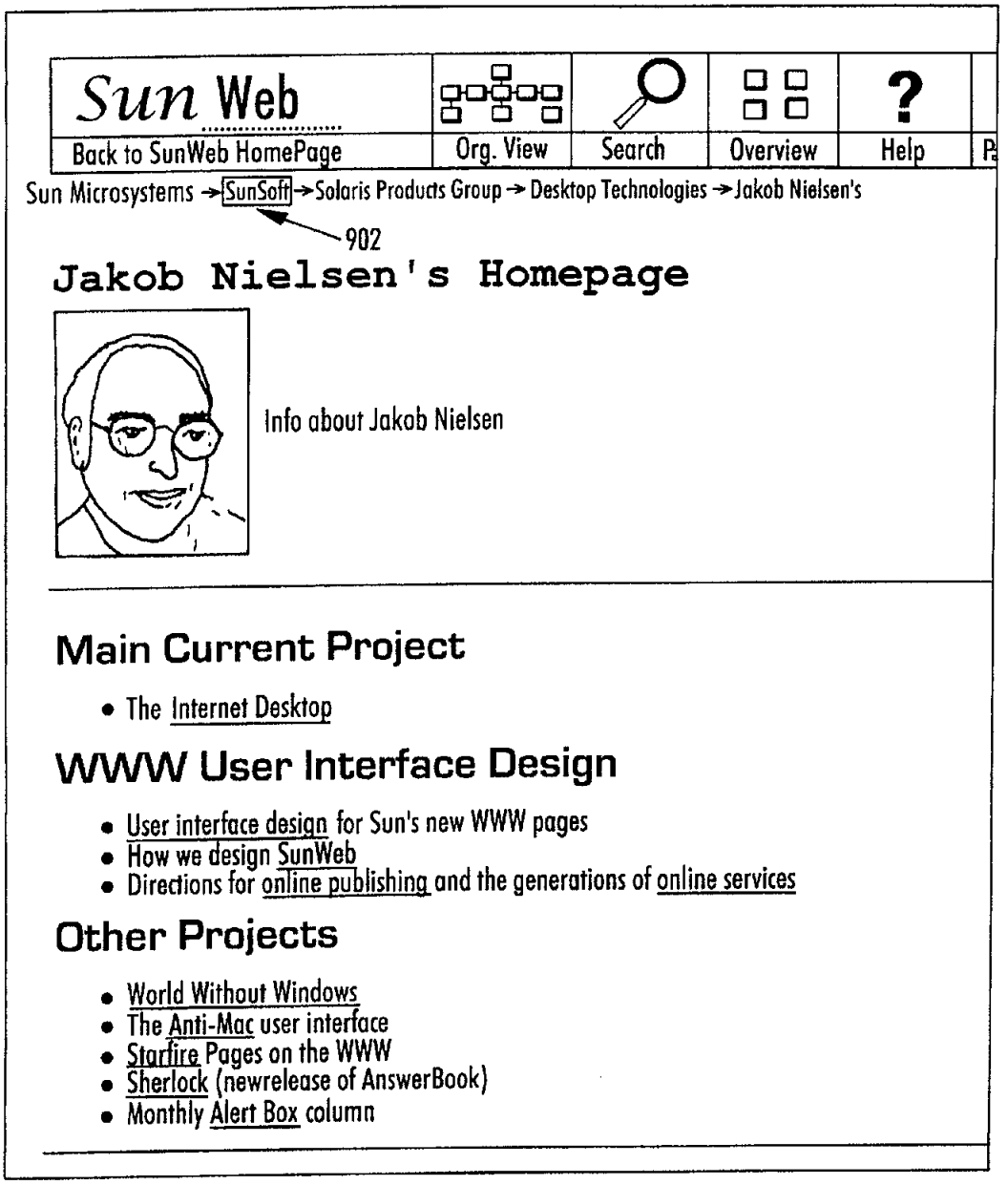

Sun Web
Back to SunWeb HomePage
Org. View
Search
Overview
Help
Sun Microsystems → SunSoft → Solaris Products Group → Desktop Technologies → Jakob Nielsen's
902
Jakob Nielsen's Homepage
Info about Jakob Nielsen
Main Current Project
The Internet Desktop
WWW User Interface Design
User interface design for Sun's new WWW pages
How we design SunWeb
Directions for online publishing and the generations of online services
Other Projects
World Without Windows
The Anti-Mac user interface
Starfire Pages on the WWW
Sherlock (newrelease of AnswerBook)
Monthly Alert Box column

Figure 9

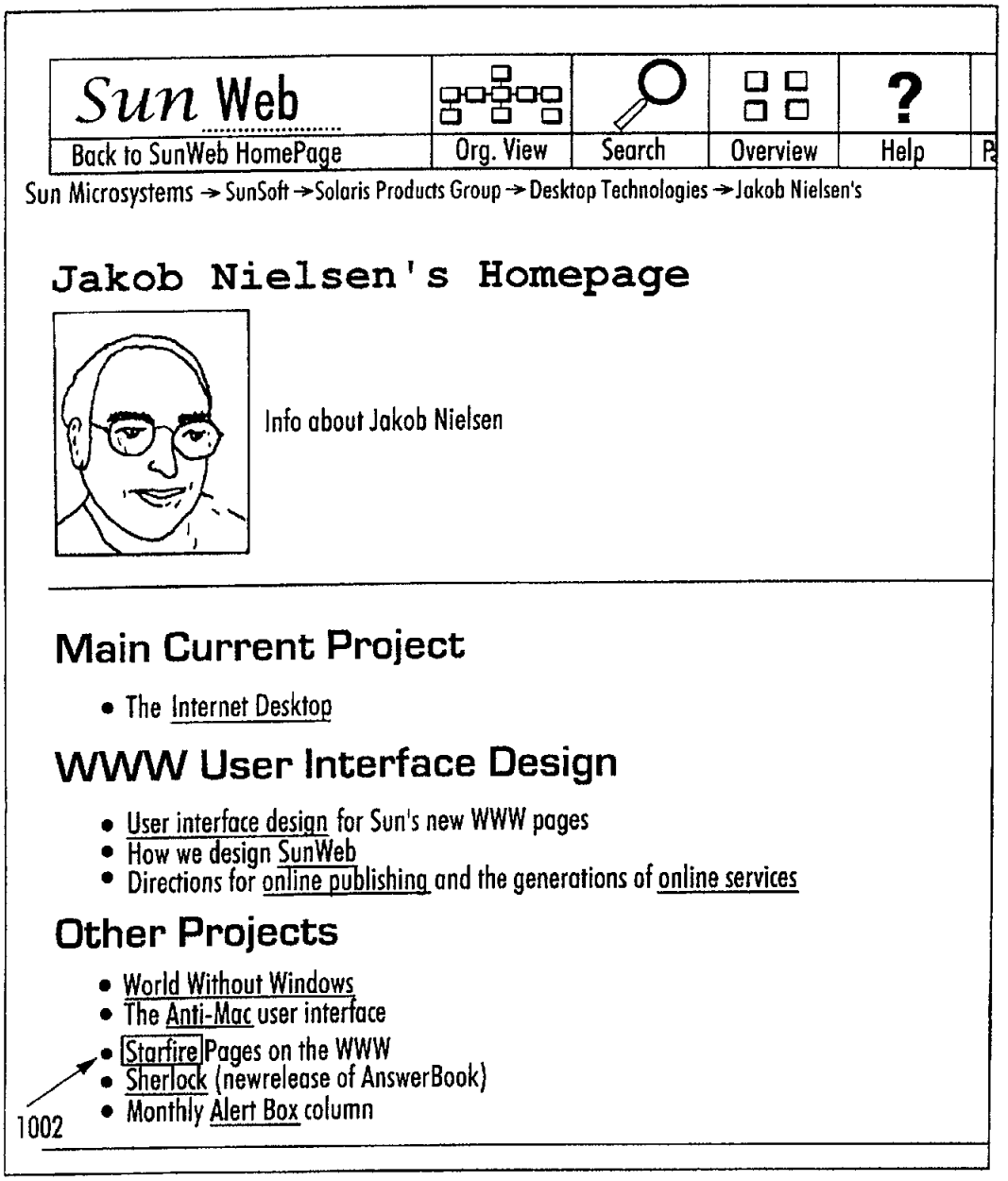

Sun Web
Back to SunWeb HomePage
Org. View
Search
Overview
Help
Sun Microsystems → SunSoft → Solaris Products Group → Desktop Technologies → Jakob Nielsen's
Jakob Nielsen's Homepage
Info about Jakob Nielsen
Main Current Project
The Internet Desktop
WWW User Interface Design
User interface design for Sun's new WWW pages
How we design SunWeb
Directions for online publishing and the generations of online services
Other Projects
World Without Windows
The Anti-Mac user interface
Starfire Pages on the WWW
Sherlock (newrelease of AnswerBook)
Monthly Alert Box column
1002

Figure 10

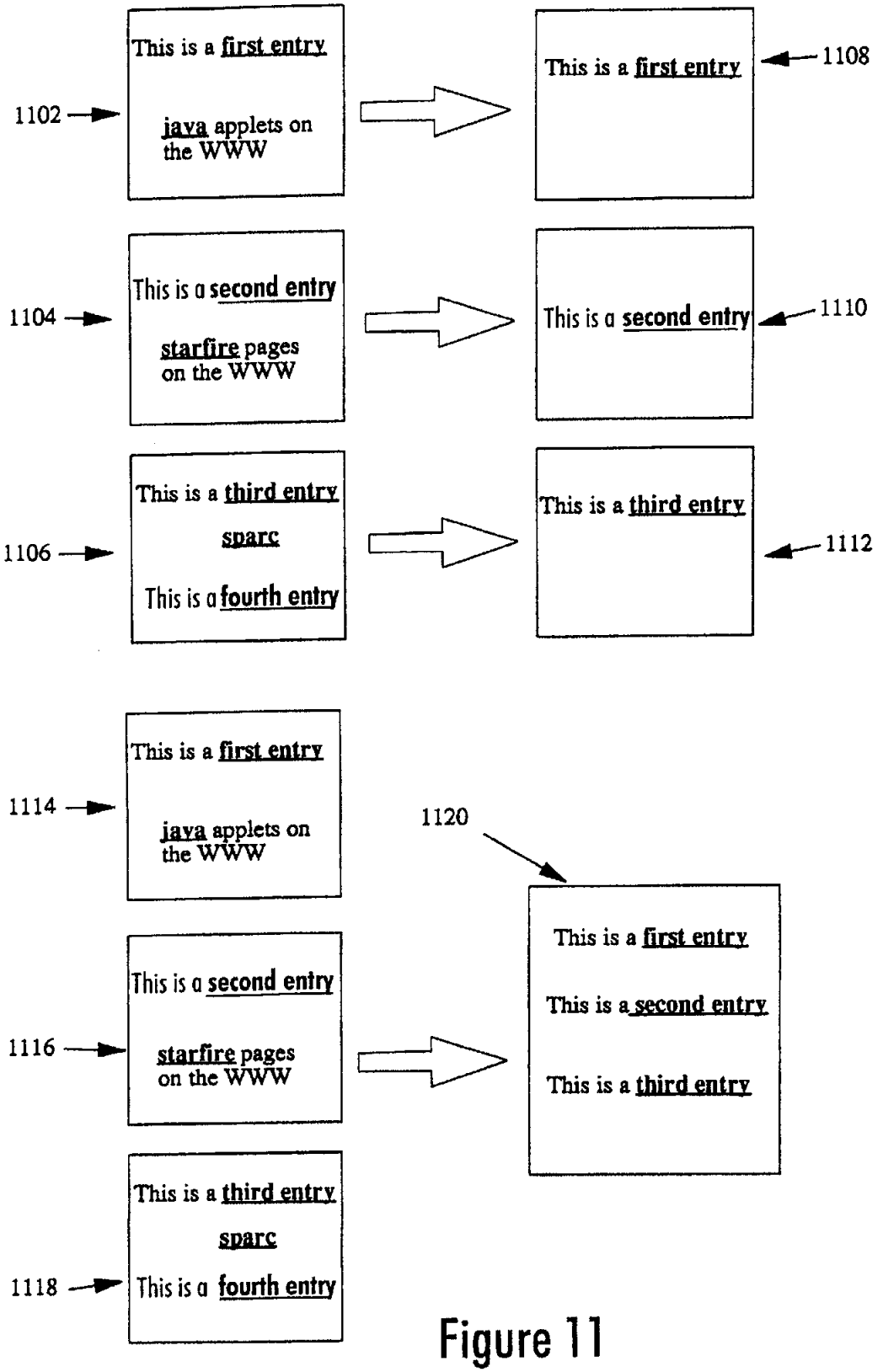

1102
This is a first entry
java applets on the WWW
1104
This is a second entry
starfire pages on the WWW
1106
This is a third entry
sparc
This is a fourth entry
1108
This is a first entry
1110
This is a second entry
1112
This is a third entry
1114
This is a first entry
java applets on the WWW
1116
This is a second entry
starfire pages on the WWW
1118
This is a third entry
sparc
This is a fourth entry
1120
This is a first entry
This is a second entry
This is a third entry

Figure 11

METHOD AND SYSTEM FOR FACILITATING ACCESS TO SELECTABLE ELEMENTS ON A GRAPHICAL USER INTERFACE

FIELD OF THE INVENTION

The present invention generally relates to graphical user interfaces (GUI) used on computer systems and, more specifically, to techniques for accessing the selectable elements in a GUI without a pointing device.

BACKGROUND OF THE INVENTION

Most computer programs executed on a computer system are controlled with a graphical user interface (GUI). The GUI displays selectable elements such as hypertext links and GUI buttons on a display device for the user to view and select. Typically, the user must use a pointing device, such as a mouse or trackball, to select the selectable elements provided by the GUI. Unfortunately, users who do not use a pointing device may be unable to use GUI based computer programs. For example, some users may have permanent physical disabilities or repetitive motion diseases, such as carpal tunnel syndrome, which prevents them from using a mouse or other pointing device. In other cases, the user may not want to use a pointing device in order to minimize the homing time associated with moving from the keyboard to the pointing device and back. It also may be necessary to eliminate the mouse from a computer system to reduce costs. In each case, these users would have great difficulty controlling a computer program which has a GUI interface. For these reasons, there is a need to access the selectable elements of a GUI without a pointing device.

Numerous conventional methods developed for selecting selectable elements on a GUI without a mouse or pointing device do not optimize a user's input speed. These methods were developed when the number of selectable elements in a given system were small and the information relatively ordered. In contrast, increasingly popular hypertext systems such as the World Wide Web (WWW) have hundreds of selectable hypertext links which are lexically unordered. As pointed out below, these conventional systems do not provide an efficient and easy method of accessing these hypertext systems without a pointing device.

FIG. 1 illustrates the technique developed by CERN for accessing a hypertext link without a pointing device. In this system, a number is assigned to each hypertext link and then added to the hypertext link text displayed to the user. A user can select a particular hypertext link by entering the hypertext link number on a keyboard device. Actuating the "enter" key takes the user to the information (e.g., a "web" page) associated with the selected hypertext link.

Unfortunately, the CERN method of accessing hypertext links is aesthetically unappealing. Adding numbers to each hypertext link changes the layout and appearance of the hypertext and impairs the designers ability to control the presentation of information on a screen. This tends to clutter the interface with unnecessary information and thus reduce the overall ease of use for the user. Furthermore, the numbers added to each hypertext link take up space which could be used to store and display additional meaningful information rather than control information used to select the hypertext link.

The CERN system is also not advantageous on a system with a large number of hypertext links. In the CERN system a user can only view and select a link being displayed on the screen. Therefore, to view the additional links a user must

scroll through numerous additional screens. Searching through a large number of hypertext links in this manner is both tedious and inefficient. For example, consider a web page on the WWW which contains a list of all companies incorporated in the United States separated by each state. Many of the companies will have several entries on the web page because they are incorporated in several different states. A user may take hours to find the place of incorporation for a particular company because he or she must scroll though multiple screens of information. Furthermore, the user will have no simple method of determining if multiple entries in the hypertext system exist for a particular company. The CERN system will not indicate if multiple occurrences of a hypertext link appear on the same web page. Thus, the user may have a difficulty determining if the company has been incorporated in multiple states.

The LYNX system is another method for accessing hyperlink information without using a pointing device. The LYNX system is shown in FIG. 2. In the LYNX system a user must move sequentially through each hypertext link using a cursor key. A user actuates the "enter" key to select the hypertext link indicated by the cursor.

The LYNX approach solves some problems found in the CERN interface and introduces additional problems of its own. The LYNX technique is an improvement over the CERN interface because no additional text must be placed in the hypertext links. However, the LYNX approach remains undesirable because it requires numerous keystrokes to locate a specific hypertext link. On a large hypertext system, the excessive number of keystrokes may deter the user from using the system. For example, assume the previously mentioned web page contained a list of approximately 1,000 companies. On the average, the LYNX system could require up to 500 keystrokes to locate a hypertext link for a particular company. This is clearly an inefficient technique for accessing hypertext or GUI system buttons without a pointing device.

The present invention provides an elegant method for selecting selectable elements on a GUI without a pointing device. The present invention is particularly well suited for hypertext systems with a large number of hypertext links such as the World Wide Web(WWW).

SUMMARY OF THE INVENTION

In one embodiment, the present invention provides a method of selecting a selectable element from a list of selectable elements on a graphical user interface (GUI) with a character based input device. The selectable element is defined as any portion of the GUI responsive to the input signals from a pointing device or a character based input device. Typically, the selectable elements include hypertext and GUI buttons on the GUI of a computer program executing on a computer. In one embodiment, the selectable element includes a character portion of data which facilitates selecting the selectable element.

In one embodiment, the method begins the selection process upon receipt of one or more characters from a character based input device. In response to receiving the characters, the method inserts each character into a match string. The match string is then compared with the character portion of at least one selectable element on the GUI. A selectable element is "armed" when a character portion of a selectable element is found which matches the match string. Arming a selectable element prepares the element for selection and typically involves highlighting, underlining or otherwise distinguishing the hypertext link from other text

on a page. The method selects the armed selectable element upon receipt of an "enter" key or an equivalent input signal from a character based input device. If a selectable element is selected then the information associated with a destination node (e.g., a web page) is displayed on a display device. Instead of selecting the armed selectable element, the method may search for a different selectable element upon receipt of additional characters from a character based input device. This particular embodiment provides an improved selection method over prior methods because it requires a minimum number of characters to quickly search and select a selectable element on a GUI.

In another embodiment, a match string timer is used to remove all the characters from the match string after a predetermined time has elapsed and the user has provided no input. This predetermined time interval begins when a user enters a first character. In one embodiment, the match string timer is loaded with a predetermined timeout value each time the method receives a new character. The match string timer begins counting down immediately after detecting the new character. If the match string timer does not count down to zero before a subsequent character is entered, the additional character is appended to the match string. However, if the match string timer counts down to zero before a subsequent character is received, the characters in the match string are removed and a subsequently entered character is inserted into an empty match string. In one embodiment, the predetermined time period is user defined and can be adjusted from one or more milliseconds up to one or more seconds of time. The match string timer allows the user to make a new search by pausing a few moments between keystrokes.

In yet another embodiment, a current location marker is used to alter the order in which one or more selectable elements are matched with the match string. Initially, the current location marker is set to the first selectable element in a list of selectable elements in the GUI. The match string is first compared with the selectable element indicated by the current location marker. If the comparison produces a match, the selectable element is armed and the current location marker is advanced to the next selectable element immediately subsequent to the armed selectable element. If the comparison does not produce a match, the current location marker is advanced to the next selectable element in preparation for another comparison. In one embodiment, the current location marker is advanced until a match is found or comparisons are made with all the selectable elements. The current location marker facilitates multiple searches on a GUI having a large number of selectable elements. Moreover, each search by the present method will begin with the last selectable element that was armed.

The present invention is faster and more powerful than the prior art because it requires a minimum number of characters to quickly search and select a selectable element on a GUI.

Notations and Nomenclature

The detailed descriptions which follow are presented largely in terms of methods and symbolic representations of operations on data bits within a computer. These method descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art.

A method is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. These steps require physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It proves convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be bourne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Useful machines for performing the operations of the present invention include general purpose digital computers or similar devices. The general purpose computer may be selectively activated or reconfigured by a computer program stored in the computer. A special purpose computer may also be used to perform the operations of the present invention. In short, use of the methods described and suggested herein is not limited to a particular computer configuration.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** illustrates a previously used technique developed by CERN for accessing a hypertext link without a pointing device.

FIG. **2** illustrates the existing LYNX technique for accessing a hypertext link without a pointing device.

FIG. **3** is a block diagram of a computer system which provides an environment for practicing one embodiment of the present invention.

FIG. **4**, illustrates a GUI system designed to display information from the World Wide Web (WWW).

FIG. **5** is a flow diagram that provides an overview of the processing that takes place when one embodiment is used to select various selectable elements without a pointing device or mouse.

FIG. **6** is a flow diagram that illustrates the steps to process a match string timer.

FIG. **7** is a flow diagram illustrating the steps taken to process GUI metacharacters.

FIG. **8** illustrates which hypertext link is highlighted on a web page from the World Wide Web when a user enters the character "S" from a keyboard device.

FIG. **9** illustrates which hypertext link is highlighted on a web page from the World Wide Web when a user enters the character "S" from a keyboard device at two separate instances.

FIG. **10** illustrates which hypertext link is highlighted on a web page from the World Wide Web when a user enters the characters "St" from a keyboard device.

FIG. **11** illustrates various embodiments of the present invention using a fast path list for storing selectable elements in a GUI.

DETAILED DESCRIPTION

Overview of an Embodiment of the Invention

In one embodiment, the present invention provides a method of selecting a selectable element from a large number of selectable elements on a graphical user interface (GUI) without a pointing device. The selectable element is defined as any portion of the GUI responsive to the input signals from a pointing device or a character based input device. Typically, these selectable elements include hypertext and GUI buttons on the GUI of a computer program executing on a computer. On most large network based systems, such as the World Wide Web (WWW), the selectable elements are not sorted lexically by title or topically according to their subject matter. Instead, data on the large network based systems remains unsorted. This vast amount of unsorted data makes accessing specific selectable ele-

ments especially difficult. The present embodiment provides an improved selection method over prior methods because it requires a minimum number of characters to quickly search and select a selectable element on a GUI.

In one embodiment the method begins the selection process upon receipt of one or more characters from a character based input device. In response to receiving the characters, the method inserts each character into a match string. The match string is then matched with the character portion of each selectable element on the GUI. A selectable element is "armed" when a character portion of a selectable element is found which matches the match string. Aiming a selectable element prepares the element for selection and typically involves highlighting, underlining or otherwise distinguishing the hypertext link from other text on a page. The method selects the armed selectable element upon receipt of an "enter" key or an equivalent input signal from a character based input device. If a selectable element is selected then the information associated with a destination node (e.g., a web page) is displayed on a display device. Instead of selecting the armed selectable element, the method may search for a different selectable element upon receipt of additional characters from a character based input device.

In another embodiment, a match string timer is used to remove all the characters from the match string after a predetermined time has elapsed and the user has provided no input. This predetermined time interval begins when a user enters a first character. In one embodiment, the match string timer is loaded with a predetermined timeout value each time the method receives a new character. The match string timer begins counting down immediately after detecting the new character. If the match string timer does not count down to zero before a subsequent character is entered, the additional character is appended to the match string. However, if the match string timer counts down to zero before a subsequent character is received, the characters in the match string are removed and a subsequently entered character is inserted into an empty match string. In one embodiment, the predetermined time period is user defined and can be adjusted from one or more milliseconds up to one or more seconds of time. The match string timer allows the user to make a new search by pausing a few moments between keystrokes.

In yet another embodiment, a current location marker is used to alter the order in which one or more selectable elements are matched with the match string. Initially, the current location marker is set to the first selectable element in a list of selectable elements in the GUI. The match string is first compared with the selectable element indicated by the current location marker. If the comparison produces a match, the selectable element is armed and the current location marker is advanced to the next selectable element immediately subsequent to the armed selectable element. If the comparison does not produce a match, the current location marker is advanced to the next selectable element in preparation for another comparison. In one embodiment, the current location marker is advanced until a match is found or comparisons are made with all the selectable elements. The current location marker facilitates multiple searches on a GUI having a large number of selectable elements. Moreover, each search using this one embodiment will begin with the last selectable element that was armed.

Embodiments of the present invention are faster and more powerful than the prior art because they require a minimum number of characters to quickly search and select a selectable element on a GUI.

Overview of One Embodiment

FIG. **3** is a block diagram of a computer system **300** which illustrates a computer system for practicing various embodiments of the present invention. Typically, a computer system **300** includes a computer **302**, a display device **304**, an input device **306** such as a keyboard, a primary storage device **308** and a secondary storage device **310**. The display device **304** displays a graphical user interface (GUI) **312** for facilitating the display of graphics and text for the user using the system **300**. Display devices **304** include, for example, printers and computer display screens such as cathode ray tubes (CRT's), light-emitting diode (LED) displays, and liquid crystal displays (LCD's). Input devices **306** can include, without limitation, electronic keyboards and pointing devices such as electronic mice, trackballs, lightpens, thumbwheels, digitizing tablets, and touch sensitive pads.

The computer **302** includes a processor **314** which fetches computer instructions from a primary storage **308** through an interface **316**, such as an input/output subsystem, connected to a bus **318**. Computer **302** can be, but is not limited to, any of the SPARCstation or Ultra workstation computer systems available from Sun Microsystems, Inc. of Mountain View, Calif., any other Macintosh computer systems based on the PowerPC processor and available from Apple Computer, Inc. of Cupertino, Calif., or any computer system compatible with the IBM PC computer systems available from International Business Machines, Corp of Somers, N.Y. which are based upon the X86 series of processors available from the Intel Corporation or compatible processors.[1] Processor **314** executes these fetched computer instructions. The processor **314** can be, but is not limited to, any of the SPARC processors available from Sun Microsystems, Inc. of Mountain View, Calif. or any processors compatible therewith. Executing these computer instructions enables the processor **314** to retrieve data or write data to the primary storage **308**, display information on one or more computer display devices **304**, receive command signals from one or more input devices **306**, or transfer data to secondary storage **310** or even other computer system which collectively form a computer network (not shown). Those skilled in the art understand that primary storage **308** and secondary storage **310** can include any type of computer storage including, without limitation, randomly accessible memory (RAM), read-only-memory (ROM), application specific integrated circuits (ASIC) and storage devices which include magnetic and optical storage media such as CD-ROM.

1. Sun, Sun Microsystems, the Sun logo, SPARC, and Ultra are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States or other countries.

The primary storage **308** stores a number of items including a GUI program **320** and an runtime environment **322**. The runtime environment **322** typically is an operating system which manages computer resources, such as memory, disk or processor time, required for the present invention to run. The runtime environment **322** may also be a microkernel, a message passing system, one or more dynamic loadable linkable modules, a browser application for the World Wide Web, a runtime interpreter environment, or any other system which manages computer resources.

The World Wide Web

The exemplary embodiment described below in accordance with FIGS. **5–11** selects selectable elements, such as hypertext links, on the World Wide Web (WWW) without a pointer device. As such, it is useful to understand how the WWW is constructed and how one embodiment of the present invention could be used to select a selectable element on the WWW.

FIG. **4** illustrates a typical GUI based system **400** designed to display information located on the WWW **404**. The WWW **404** is a warehouse for millions and millions of bytes of data contained in discrete storage elements called web pages. A web page **406** contains data **408** and a layout language **410** which describes how the data should be displayed. The layout language **410** used within each web page, for example, includes any combination of Java™, JavaScript™, VRML, hypertext markup language (HTML) or any other language which supports hypertext functionality.[1] The user views a web page from the World Wide Web using a particular type of GUI program called a browser. A browser **402** accepts the data **408** and the layout language **410** and in response produces a graphical image containing hypertext links along with text, graphics and various other types of multimedia.

1. Sun, Sun Microsystems, the Sun logo, Java, and Javascript are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States or other countries.

The hypertext links are of particular importance because they facilitate access to web pages located all over the WWW. An anchor tag in HTML is used to define a hypertext link on the WWW. The anchor tag has a destination portion and a text portion. Furthermore, the anchor tag is delimited at the beginning by a "<A>" and at the end by a "<A>" character sequence. The destination portion contains an entry which indicates the location of the destination web page. The text portion is the text which appears on the GUI and is associated with the hypertext link. For example, consider a hypertext link having the following HTML anchor tag:

<A HREF=http://sunweb.ebay/sunweb/orgs>Sun Microsystems</A>

The destination portion of this tag is "http://sunweb.ebay/sunweb/orgs." The text "Sun Microsystems" occupies the text portion of this HTML based hypertext link. In an exemplary embodiment, the text portion of the anchor tag is utilized by the present invention, as described in detail later herein, to facilitate searching and selecting hypertext links on the WWW without a pointing device.

The present invention is not limited for use with GUI systems designed for the WWW. One skilled in the art could apply the teachings and suggestions of the present invention to any GUI system which has text associated with selectable elements. Essentially, selectable elements includes all portions of the GUI which are responsive to the input signals from a pointing device or a character based input device.

Exemplary Steps of One Embodiment

FIGS. **5–11** illustrates one method for practicing the present invention. The flow diagrams described herein broadly illustrate the logical flow of steps to perform one embodiment of the present invention. Accordingly, numerous steps may be added to, or taken away from the flow diagrams, without departing from the scope of the invention. Furthermore, the order of execution of the steps in the flow diagrams may be changed without departing from the scope of the invention. Additional considerations in implementing the method described by the flow diagrams may also dictate changes in the selection and order of the steps.

FIG. **5** is a flow diagram that provides an overview of the processing that takes place in the GUI **402** when an embodiment of the present invention is used to select various selectable elements without a pointing device or mouse. Before processing begins, the present embodiment initially requires a list of selectable elements to select from. Typically, a list of selectable elements is generated from data provided by an underlying application using the GUI program. For example, on the World Wide Web (WWW) a web page contains a list of anchor tags which describe hypertext links in HTML. A browser processes these anchor tags and displays corresponding hypertext links to the user on a display device. This embodiment receives the text portion of these anchor tags and creates a selectable element list for further processing described below.

Generally, selecting a selectable element requires several steps. First, the user indicates to the method which selectable element should be armed by actuating one or more characters with a character based input device. The method then highlights the armed selectable element in preparation for selection by the user. Next, the user actuates a selection character, such as an "enter" key, indicating that the armed selectable element should be selected. In response to receiving the selection character, the method selects the armed selectable element. Generally, selecting a selectable element causes the functionality associated with the selectable element to be performed (i.e. retrieving a "web" page).

In one embodiment, selecting the selectable element follows the steps **502–528** in FIG. **5**. At step **502**, the current location marker is set to the first selectable element in the selectable element list the first time the invention is used. The current location marker indicates which selectable element should be compared first in the matching process described below. For example, FIG. **8** shows a web page displayed on a browser containing several hypertext links as indicated by the underlined text. In an exemplary embodiment, the current location marker in FIG. **8** is initially set to the first hypertext link **802** indicated by the characters "Sun Microsystems".

At step **504** the method receives a character on a character based input device such as a keyboard. In step **506**, the method analyzes the character and determines if the character received was an "enter" key. In general, the "enter" key represents any input signal which indicates that a selectable element should be selected. If the method receives the "enter" key, the selectable element is selected as described in detail below. At step **508**, the method adds the character to a match string. The match string holds a set of characters entered by the user for use in the matching step **510**.

At step **510** character comparison techniques well known in the art are used to compare the match string with one or more selectable elements. In one embodiment, a sequence of one or more characters in the match string is compared to a sequence of one or more characters associated with one or more selectable elements. The method begins the comparisons starting with the selectable element indicated by the current location marker.

Steps **516–520** prevent two selectable elements from being armed simultaneously. Initially, a selectable element may be armed from a previous attempt to select a selectable element. This occurs when the method arms a selectable element without subsequently selecting it. At step **516**, the method detects if there is a previously armed selectable element and a new selectable element is about to be armed. In one embodiment, if a selectable element was armed from a previous matching step, the method will disarm the previously armed selectable element at step **518** before the new matching selectable element is armed at step **520**. If no selectable element was previously armed, the method will bypass step **518** and arm the matching selectable element at step **520** directly. Generally, the selectable element is armed when the matching step determines the characters within the match string match the text portion of the selectable element. Arming a selectable element includes highlighting, underlining or otherwise distinguishing the text of the selectable element from other text within the current page being displayed to the user.

FIG. **8** and FIG. **10** will be used to illustrate how a previously armed selectable element is disarmed before a new selectable element is armed. Assume a user has entered the character "S" with a keyboard device and caused the hypertext link **802** to become armed. At this point, the current location marker is pointing to the hypertext link immediately subsequent to hypertext link **802**. Before selecting hypertext **802**, however, the user then enters the letter "t" using the keyboard. The method adds the letter "t" to the match string (step **508**) and proceeds to compare the match string with the subsequent selectable elements starting with the current location marker. The method indicates a match between the match string and a hypertext link **1002** (step **510**). As a result, step **518** disarms hypertext link **802** in FIG. **8** containing the text "Sun Microsystems" and arms the hypertext link **1002** in FIG. **10** containing the text "Starfire". In this way, only one selectable element is armed at any given moment and the user is not distracted by having multiple selectable elements to choose from. Also, the previously armed selectable element remains visibly armed on the display unit up until a new selectable element is armed.

Referring back to FIG. **5**, at step **528** the current location marker is advanced to the first selectable element subsequent to the matching selectable element. This causes the next search to occur subsequent to the matching selectable element. For example, in FIG. **8** the current location marker is initially set to a hypertext link **802** because it is the first selectable element on the web page. Assume the method receives the letter "S" from a character based input device. The method receives the letter "S" and places it in the match string (step **508**). Next, the matching step compares the "S" in the match string with the character portion of one or more hypertext link in the current web page until a match is found (step **510**). In this case, the matching step **510** arms the hypertext link **802** in FIG. **8** because the character "S" in the match string matches the character "S" in "Sun Microsystems". Moreover, the method advances the current location marker to the first selectable element subsequent to the hypertext link **802** (e.g. the hypertext link containing the text "SunSoft"). Now assume the user removes the letter "S" from the match string by typing a backspace and then enters a new letter "S" into the match string. In this case, the method arms hypertext link **902** in FIG. **9**, in a manner described above, because the hypertext link **902** pointed to by the current location marker matches the match string.

In some cases, the method will receive a sequence of characters from a character based input device which do not match any selectable element in the GUI. The user entering these characters may be searching for selectable elements which do not exist on the particular GUI being displayed. This portion of the invention handles the case where no selectable element matches the characters in the match string. Referring back to FIG. **5**, at step **510** the method determines if the characters in the string do not match a selectable element. In one embodiment, if the selectable element has already been armed (step **512**) the method will indicate an error (step **514**). Those of ordinary skill will understand that the error indication provided by the method includes audio feedback, visual feedback or any other feedback capable of indicating an error to the user. The error indicates that the additional characters received by the method do not match another selectable element on the GUI. Consequently, the method may already have armed the selectable element which most closely matches the characters in the match string. In contrast, if no selectable element has been armed the method will return to step **504** to receive additional characters from the user and will not make an error indication. In this case, the method has not found a selectable element which matches the characters in the match string.

After steps **512**, **514**, or **528**, the method returns to step **504** to receive an additional character input from the user. This portion of the processing determines whether a selectable element should be armed or the armed selectable element should be selected. At step **506**, if an additional character, other than the "enter" key or an equivalent input signal, is provided, the method will add the character to the match string (step **508**) and arm a selectable element in accordance with steps **510**–**528**. Alternatively at step **506**, if the method receives the "enter" key, or an equivalent input signal which indicates that a selectable element should be selected, the method first determines if a selectable element is currently armed (step **522**). If a selectable element is currently armed (step **522**), the method selects the armed selectable element (step **524**). When the method selects the selectable element (step **524**), the functionality associated with the selectable element is performed. For example, FIG. **8** illustrates a previously armed hypertext link **802** containing the words "Sun Microsystems". Upon receipt of the "enter" key, an exemplary embodiment selects hypertext link **802**, retrieves the web page for "Sun Microsystems" and displays the information associated with the web page on display device.

If the method receives the "enter" key and no selectable element is currently armed then, at step **526**, the system will generate an error message for the user. This error indicates that the user of the method has attempted to select a selectable element which has not been armed. At step **526**, various embodiments can be used to indicate that a user is attempting to select a selectable element before it has been armed. In one embodiment, an audible message is provided when a user tries to select a selectable element before a selectable element has been armed. In an alternative embodiment, a visual image or message is provided to the user trying to select a selectable element before a selectable element has been armed. For example, the method may provide an audible beep the first and second time the user actuates the "enter" key without a selectable element first being armed. The third time, however, a window may appear indicating that the user has attempted to select a selectable element before an element has been armed. Those skilled in the art understand that various combinations of audio and visual messages may be provided to ensure the user realizes an error has been made.

The Match String Timer

In an alternative embodiment, a match string timer is used to remove characters stored in the match string at step **508**. FIG. **6** is a flow diagram that illustrates the steps taken to process the match string using a match string timer. At step **602**, an embodiment of the present invention receives a character from the user. At step **604**, if the match string timer counts down to zero and times out the method removes all characters from the match string (step **606**). However, if the match string timer has not timed out (step **604**) then the method inserts the character into the match string (step **608**). Each time a character is received the system loads the match string timer with another predetermined timer value (step **610**). Immediately after loading the match string timer the match string timer begins a new countdown process (step **612**). The match string timer ensures that a new match string is used at step **508** for comparisons when a user pauses a predetermined time interval between key strokes.

GUI Metacharacters

In yet another alternative embodiment, the method receives a GUI metacharacter from a character based input device which controls the operation of the GUI. In one embodiment, these special keystrokes are reserved for use as GUI metacharacters and are not compared with the text of the selectable elements within the GUI. GUI metacharacters are advantageous in a mouseless environment because they provide an efficient method of controlling the GUI without a pointer device. In one embodiment, the processing of GUI metacharacters occurs after step **506** and before either step **504** or step **508**. FIG. **7** illustrates the processing of these GUI metacharacters in detail.

Referring to FIG. **7**, a flow diagram outlines the method in which several GUI metacharacters are processed. The method performs steps **704–718** to process GUI metacharacter. At step **704**, the method determines if the user provided a screen scroll GUI metacharacter. At step **706**, the method causes a screen of information on the GUI to scroll up upon receipt of an up-screen scroll character (not shown). At step **706**, the method can also cause a screen of information on the GUI to scroll down upon receipt of a down-screen scroll character (not shown). In one embodiment, the method sets the current location marker to the first selectable element which appears in the screen each time the GUI window scrolls up -or down to a new set of selectable elements.

At step **708**, the method determines if the user provided a fast path store keystroke. The fast path list is a collection of often used or desirable selectable elements on the GUI. The method stores the armed selectable element to be stored in a fast path list upon receipt of a fast path store keystroke character. Later, the method provides the fast path list of selectable elements to the user for additional searching. In one embodiment, the fast path list of selectable elements is displayed in a "pop-up" type window on the GUI. The user searches the fast path list by entering characters from a character based input device. In response, the method receives these characters and searches through the fast path list in a manner similar to the one described in steps **502–528**.

FIG. **11** illustrates various embodiments of fast path lists for storing selectable elements in a GUI. In one embodiment, selectable elements from GUI pages **1102**, **1104**, and **1106** are stored in separate fast path lists **1108**, **1110**, and **1112** respectively. This allows the user to segregate important hypertext links from each web page in separate lists. In yet another embodiment, selectable elements retrieved from GUI pages **1114**, **1116**, and **1118** are stored in a single fast path list **1120**. In this case, the user can consolidate all important hypertext links in a central location for easy access. Those skilled in the art understand that the selectable elements in the fast path list may be deleted by user using techniques well known in the art. Furthermore, the Applicant contemplates that the hypertext links contained within the fast path lists in either embodiment can be sorted lexically, chronologically by topic, or even by frequency of use.

Referring back to FIG. **7**, at step **712** the method may also determine if the user provided a clear match string GUI metacharacter. In response to receiving the clear match string GUI metacharacter, the method removes all characters in the match string (step **714**). This enables the user to quickly remove the characters in the match string and begin searching for a new selectable element in the GUI.

In general, GUI metacharacters are designed to make searching and selecting a selectable element without a pointing device a more efficient process. In one embodiment, if a combination of GUI metacharacters arms a selectable element, the current location marker is advanced to the next subsequent selectable element. This enables the user to advance the current location marker using one or more GUI metacharacters and then begin searching subsequent to the current location marker's new location. For example, a user enters several "tab" keystrokes to advance to a particular hypertext link on a web page. In one embodiment, the particular hypertext link is armed and the current location marker is advanced to the next subsequent hypertext link. In an alternative embodiment, if the current location marker is not advanced, subsequent searches begin from the current location marker's original location.

Those of ordinary skill will understand that many other GUI metacharacters could be provided by embodiments of the present invention to control the operation of the GUI. The exemplary uses of GUI metacharacters above are provided as examples and should not limit the scope of the present invention.

Using the methods illustrated in FIGS. **5–11** enables a user to select selectable elements in a graphical user interface without the use of a mouse or pointing device. These methods are particularly useful for searching and selecting hypertext links from a large hypertext system containing lexically unordered data. The teachings and suggestions of the present invention allow a user to quickly navigate to an appropriate hypertext link by entering characters associated with the appropriate hypertext link.

Embodiments of present invention provide a number of advantages over the methods used in the prior art. For example, embodiments of the present invention do not interfere with the layout and placement of selectable elements in a GUI. In the prior art, some methods required the addition of numeric text to each selectable element. These additions destroyed the original layout intended by the original hypertext designers and were visually unappealing. Embodiments of the present invention require no additional text or modification for the hypertext selection to work.

In addition, embodiments of the present invention require less keystrokes to search for a particular selectable element in a GUI based system than the prior art. Some prior art systems required a user to page through each selectable element one element at a time using an up or down keystroke on the keyboard. Getting to the last selectable element in a list required scrolling through each element. In contrast, embodiments of the present invention allow one to quickly search for the proper selectable element by entering only a few keystrokes. The user is quickly taken to the selectable element which matches the characters entered by the user. This is particularly useful in searching for hypertext links on the World Wide Web where there are many hypertext links which do not appear in alphabetical, chronological or numeric order.

While specific embodiments have been described herein for purposes of illustration, various modifications may be made without departing from the spirit and scope of the invention. Various embodiments of the present invention can be implemented on the World Wide Web (WWW) or any other GUI based system which utilizes similar selectable elements. These various embodiments are particularly useful on GUI based systems where the selectable elements are lexically unordered and too numerous to simultaneously display on a single display screen. Accordingly, the invention is not limited to the above described embodiments, but instead is defined by the appended claims in light of their full scope of equivalents.

What is claimed is:

1. A method for facilitating the selection of at least one selectable element from a multiplicity of lexically unordered selectable elements on a graphical user interface (GUI) having a plurality of elements, at least one of the multiplicity of selectable elements having a character portion, said method comprising the computer implemented steps of:

receiving one or more characters from a character based input keyboard device;

comparing said one or more characters received from said character based input keyboard device with said character portion from one or more of said multiplicity of lexically unordered selectable elements; and

matching said one or more characters received from said character based input keyboard device with said character portion of at least one selectable element within said multiplicity of lexically unordered selectable elements.

2. The method of claim 1 further comprising the step of:

arming a selectable element which matched said one or more characters received from said character based input keyboard device.

3. The method in claim 2 wherein said arming step further comprises the step of disarming a previously armed selectable element before arming said selectable element which matched said one or more characters received from said character based input keyboard device.

4. The method of claim 2 further comprising the step of:

selecting said armed selectable element in response to receiving an actuation input signal which indicates said armed selectable element should be selected.

5. The method of claim 1 further comprising the step of:

setting a current location marker to a predetermined selectable element in said multiplicity of lexically unordered selectable elements, wherein said current location marker indicates which selectable element the comparing step should begin comparisons with.

6. The method in claim 5 wherein said predetermined selectable element is said first selectable element in said multiplicity of lexically unordered selectable elements.

7. The method in claim 5 further comprising the step of advancing said current location marker to said selectable element immediately subsequent to a selectable element which matches said one or more characters received from said character based input keyboard device.

8. The method in claim 1 further comprising the step of receiving a GUI metacharacter from said character based input keyboard device, wherein said GUI metacharacter controls a predetermined function of the GUI.

9. The method in claim 8 further comprising the steps of:

receiving a fast path store GUI metacharacter from said character based input keyboard device; and

in response to receiving said fast path store GUI metacharacter, storing an armed selectable element in a fast path selection list, wherein said selectable element is stored within said fast path selection list for rapid selection at a subsequent time period.

10. The method in claim 1 wherein the selectable elements are hypertext links.

11. The method in claim 1 wherein the selectable elements are GUI buttons.

12. An apparatus for facilitating the use of a character input keyboard device in the selection of at least one selectable element from a multiplicity of lexically unordered selectable elements on a graphical user interface (GUI) program having a plurality of elements, at least one of said multiplicity of selectable elements having a character portion, said GUI executing on a computer system including a processor, a memory, a display device displaying the graphical user interface (GUI) the character input keyboard device, a storage device, and a mechanism for communicating between the processor, the character input keyboard device and the memory, the memory including the graphical user interface (GUI) program and a runtime system, said apparatus comprising:

a receiver module configured to receive one or more characters from a character based input keyboard device;

a comparison module configured to compare one or more characters received from the character based input keyboard device with said character portion from one or more of said multiplicity of lexically unordered selectable elements; and

a match module configured to match said one or more characters received from said character based input keyboard device with said character portion of at least once selectable element within said multiplicity of lexically unordered selectable elements.

13. The apparatus of claim 12 further comprising:

an arming module configured to arm a selectable element which matched said one or more characters received from said character based input keyboard device; and

a selection module configured to select said armed selectable element in response to receiving an actuation input signal.

14. The apparatus in claim 13 further comprising a current location marker set to a predetermined selectable element in said multiplicity of lexically unordered selectable elements, wherein said current location indicate which selected element said comparison module should begin comparisons with.

15. The apparatus in claim 14 where said current location marker is advanced to said selectable element subsequent to said selectable element which matches said one or more characters received from said character based input keyboard device.

16. The apparatus in claim 12 wherein the selectable elements are hypertext links.

17. The apparatus in claim 12 wherein the selectable elements are GUI buttons.

18. The apparatus in claim 12 further comprising a receiver module for receiving one or more GUI metacharacters from a character based input keyboard device, wherein a GUI metacharacter controls a predetermined function of the GUI.

19. The apparatus in claim 12 wherein said computer system is a client computer coupled to one or more server computer systems over a communications link and wherein data is exchanged between said client computer system and at least one server computer system over said communications link.

20. The apparatus in claim 12 wherein said computer system is a server computer coupled to one or more client computer systems over a communications link and wherein data is exchanged between said server computer system and at least one client computer system over said communications link.

21. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine for facilitating the selection of at least one selectable element from a multiplicity of lexically unordered selectable elements on a graphical user interface (GUI), at least one selectable element within said multiplicity of lexically unordered selectable elements having a character portion, said program comprising:

code that receives one or more characters from a character input keyboard device;

code that compares one or more characters received from said character input keyboard device with said character portion from one or more of said multiplicity of lexically unordered selectable elements; and

code that matches said one or more characters received from said character based input keyboard device with at least one selectable element within said multiplicity of lexically unordered selectable elements.

22. The storage device of claim 21 further comprising:

code that arms a selectable element which matched said one or more characters received from said character based input keyboard device.

23. The storage device of claim 22 further comprising:

code that selects the armed selectable element in response to receiving an actuation input signal.

24. The storage device in claim 21 wherein the selectable elements are hypertext links.

25. The storage device in claim 21 wherein the selectable elements are GUI buttons.

26. The storage device in claim 22 further comprising code capable of receiving one or more GUI metacharacters from a character based input keyboard device, wherein said GUI metacharacter controls a predetermined function of the GUI.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 5,963,950
DATED : Oct. 5, 1999
INVENTOR(S) : Nielsen et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In the Claims, in Claim 14, line 4, after "location" insert word --**marker**--, further in the same line change "indicate" to --**indicates**-- and after word "which" change "selected" to --**selectable**--.

Signed and Sealed this

Twenty-fifth Day of April, 2000

Attest:

Q. TODD DICKINSON

Attesting Officer

Director of Patents and Trademarks

EXHIBIT D

US005761662A

United States Patent [19]

Dasan

[11] Patent Number: 5,761,662

[45] Date of Patent: Jun. 2, 1998

[54] **PERSONALIZED INFORMATION RETRIEVAL USING USER-DEFINED PROFILE**

[75] Inventor: **Vasanthan S. Dasan**, Colorado Springs, Colo.

[73] Assignee: **Sun Microsystems, Inc.**, Mountain View, Calif.

[21] Appl. No.: **852,455**

[22] Filed: **May 8, 1997**

Related U.S. Application Data

[63] Continuation of Ser. No. 360,997, Dec. 20, 1994, abandoned.

[51] **Int. Cl.**[6] **G06F 17/30**

[52] **U.S. Cl.** **707/10**; 707/9; 707/100; 707/104; 707/3; 707/4; 707/513; 395/200.36

[58] **Field of Search** 707/10, 9, 3, 4, 707/100, 104, 513; 395/200.36

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,124,909	6/1992	Blakely et al.	395/200
5,200,993	4/1993	Wheeler et al.	379/96
5,309,351	5/1994	McCain et al.	364/132
5,347,632	9/1994	Filepp et al.	395/200
5,404,523	4/1995	DellaFera et al.	395/650
5,463,625	10/1995	Yasrebi	370/85.13
5,550,657	8/1996	Bly et al.	395/425

OTHER PUBLICATIONS

M Sanderson, "NRT (News Retrieval Tool) A User's Guide," IEEE Colloq. 1990, No. 101; pp. 5/1–5/14.

MCI Telecommunications, Inc., *Network MCI Business*, Pamphlet, Marketing Information (Oct. 1994).

Berners–Lee, T. and Daniel Connolly, *Hypertext Markup Language*, 1–36, Jun. 1993.

Berners–Lee, T., *Hypertext Transfer Protocol*, 1–25, Jun. 1993.

Author Unknown, *A Beginner's Guide to HTML*, 1–14, Jun. 1993.

M. Pazzani et al.: Learning from Hotlists and Coldlists: Towards a WWW Information Filtering and Seeking Agent; pp. 492–495; Dept. of Information and Computer Science, UC Irvine, Nov. 1995.

J. Powell: Adventures with the World–Wide Web: Creating a Hypertext Library Information System; pp. 59–66, Feb. 1994.

T. Lau: Building a Hypermedia Information System on the Internet; pp. 192–197, Sep. 1994.

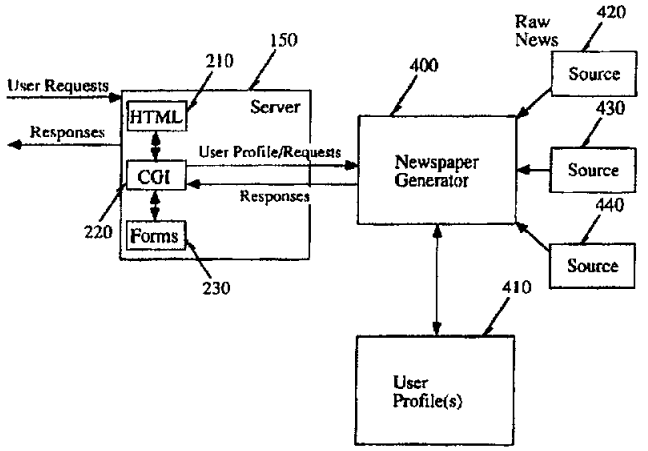
B.K. Duval et al.: Microcomputer Applications in the Library; pp. 269–279, 1994.

Primary Examiner—Thomas G. Black
Assistant Examiner—C. Lewis
Attorney, Agent, or Firm—Blakely Sokoloff Taylor & Zafman

[57] **ABSTRACT**

An automatic method and system for retrieving information based on a user-defined profile (e.g. a personalized newspaper). A user-controlled client establishes communication with a stateless server, the server presenting a list of options to the client between the server and the client. The client provides an identification of the user-defined profile. The server engages a first application program, the first application program retrieving the user-defined profile wherein the user-defined profile identifies information which is of interest to the user. The first application program examines a database of information and automatically retrieves a subset of the information from the database based upon which information is of interest to the user as identified in the user-defined profile. The server presents the subset of the information from the database as generated by the first application program to the client. The first application program can store a file containing the user-defined profile in order to retain a state of the user-profile, and cause the stateless server to emulate a server which retains its state from session to session.

26 Claims, 12 Drawing Sheets



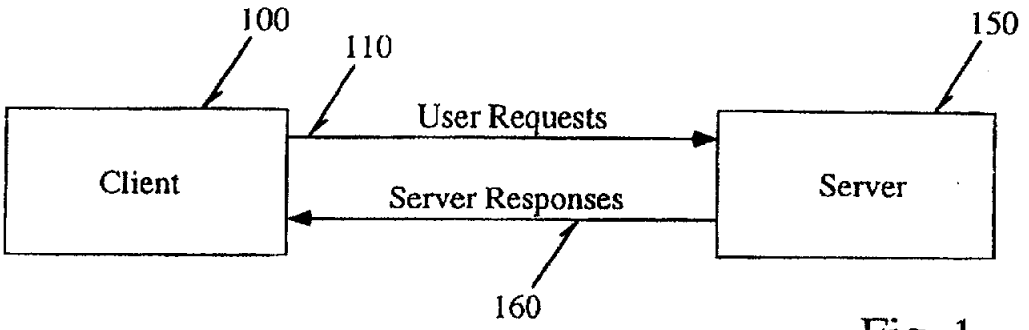

100
110
150
User Requests
Client
Server
Server Responses
160

Fig. 1

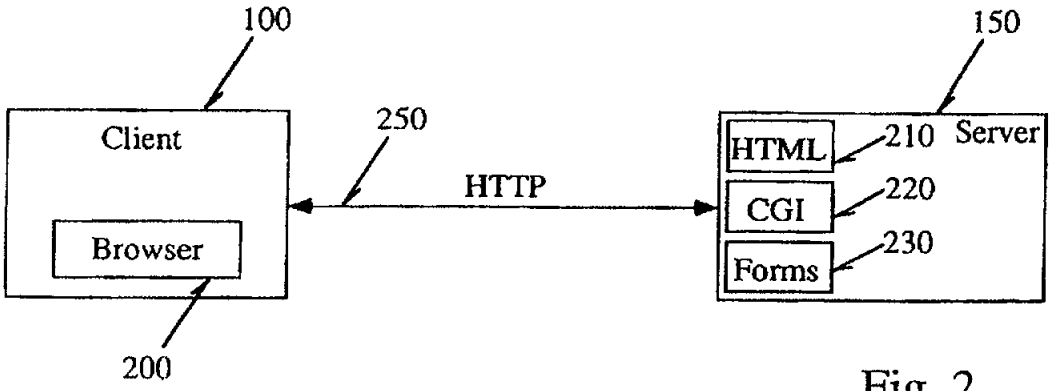

100
150
Client
250
HTML
210
Server
HTTP
CGI
220
Browser
230
Forms
200

Fig. 2

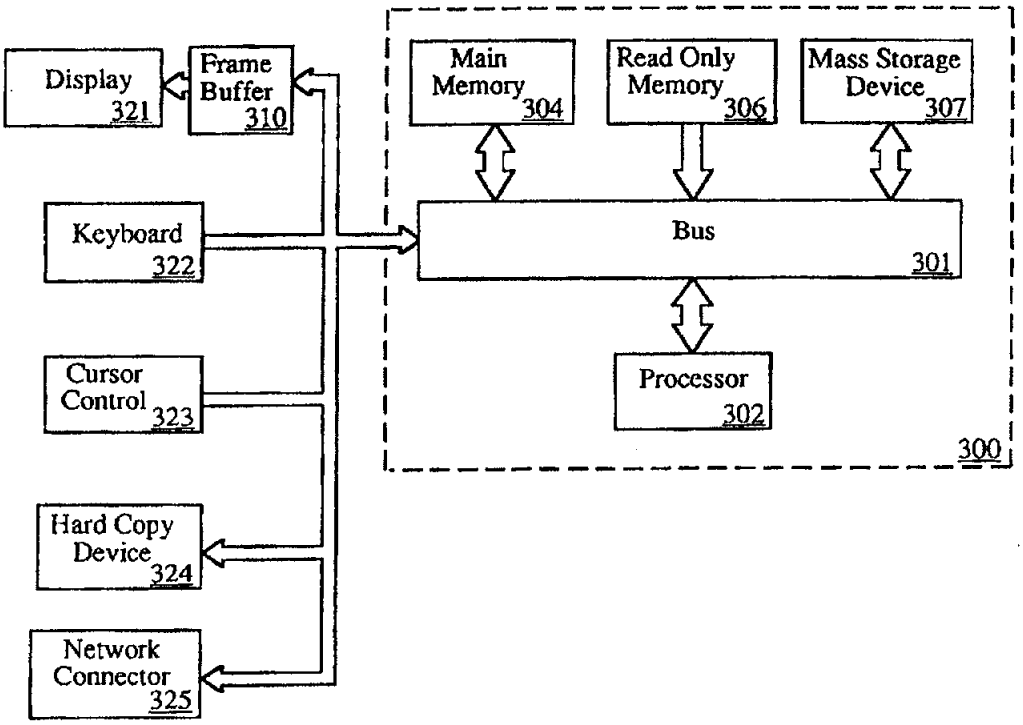

Display
321
Frame
Buffer
310
Main
Memory
304
Read Only
Memory
306
Mass Storage
Device
307
Keyboard
322
Bus
301
Cursor
Control
323
Processor
302
300
Hard Copy
Device
324
Network
Connector
325

Fig. 3

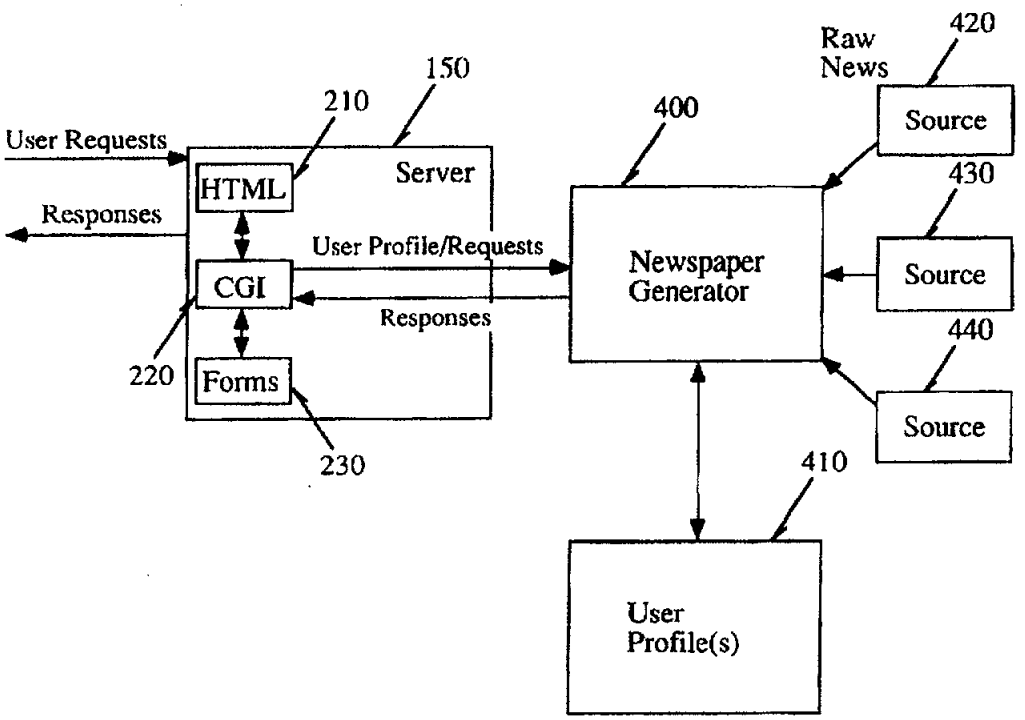

User Requests
Responses
150
210
HTML
Server
CGI
Forms
220
230
User Profile/Requests
Responses
400
Newspaper Generator
Raw News
420
Source
430
Source
440
Source
410
User Profile(s)

Fig. 4

Fig. 5a

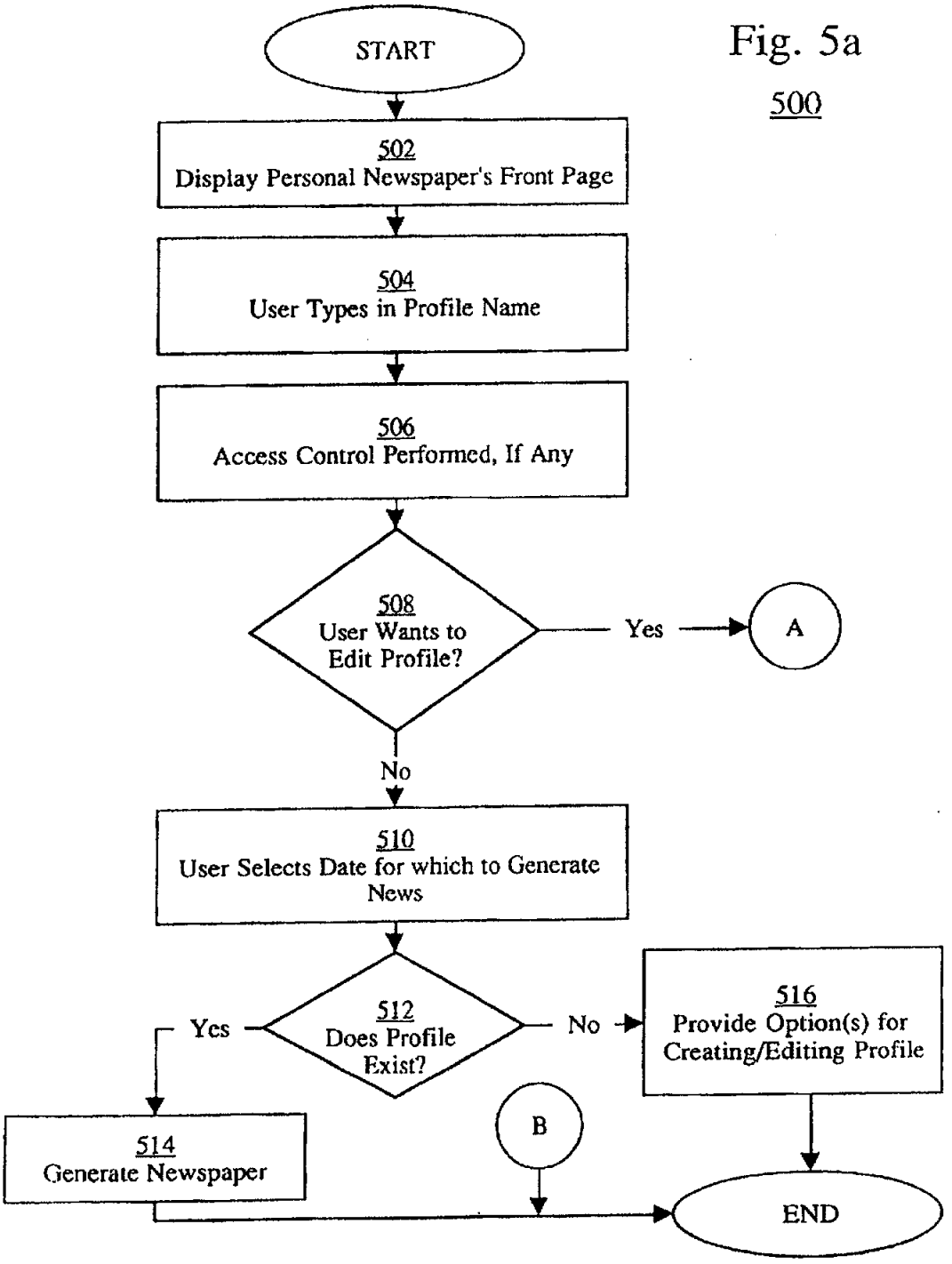
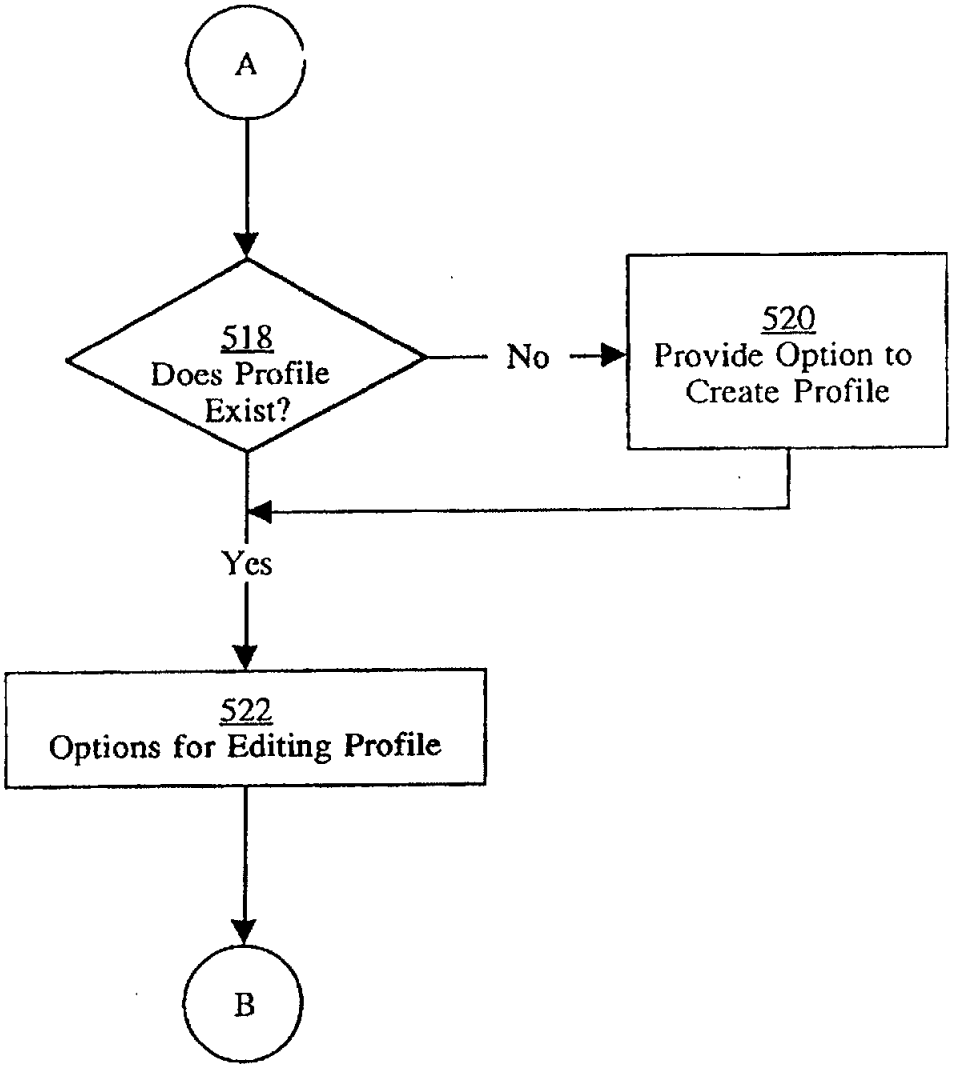

500
START
502
Display Personal Newspaper's Front Page
504
User Types in Profile Name
506
Access Control Performed, If Any
508
User Wants to Edit Profile?
Yes
A
No
510
User Selects Date for which to Generate News
512
Does Profile Exist?
Yes
No
516
Provide Option(s) for Creating/Editing Profile
514
Generate Newspaper
B
END

Fig. 5b

500


A
518
Does Profile Exist?
No
520
Provide Option to Create Profile
Yes
522
Options for Editing Profile
B

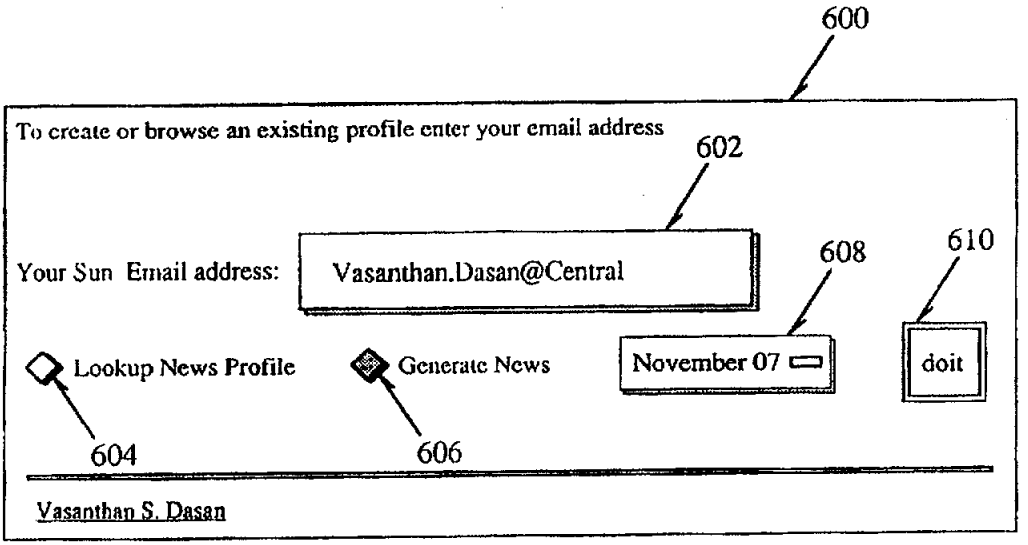

600
To create or browse an existing profile enter your email address
602
Your Sun Email address:
Vasanthan.Dasan@Central
608
610
Lookup News Profile
Generate News
November 07
doit
604
606
Vasanthan S. Dasan

Fig. 6

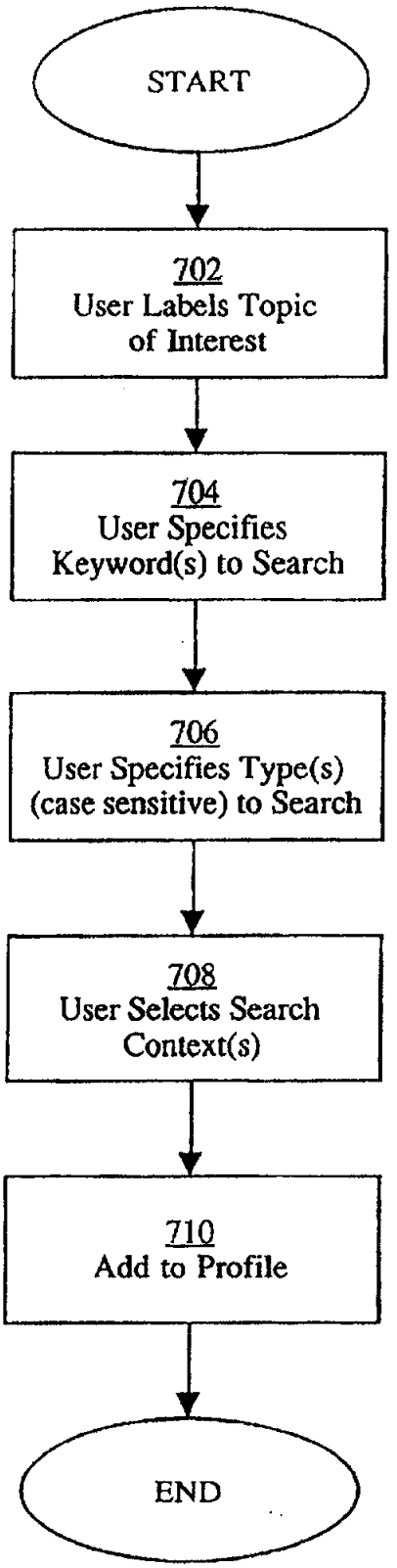


Fig. 7

Add to Profile

700

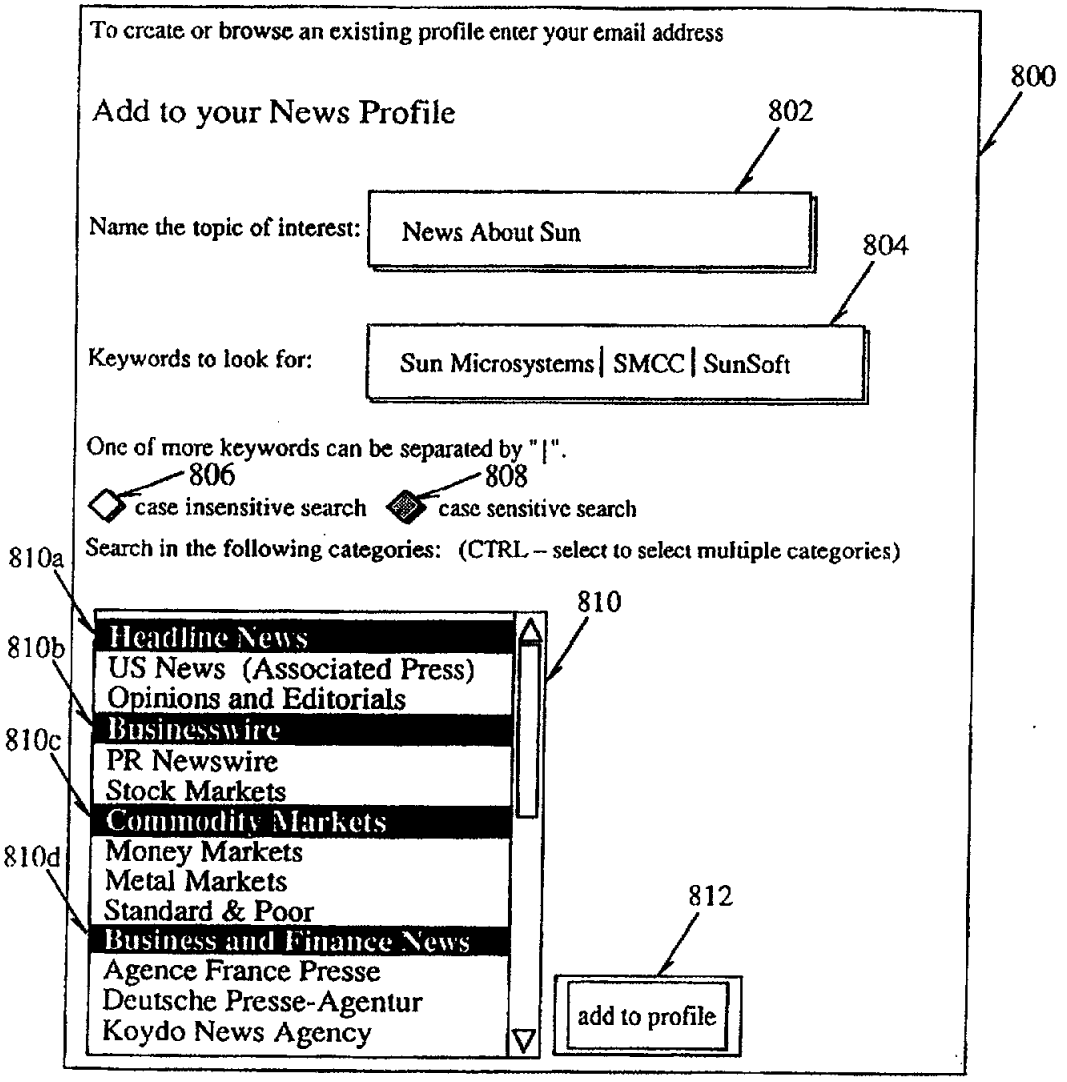
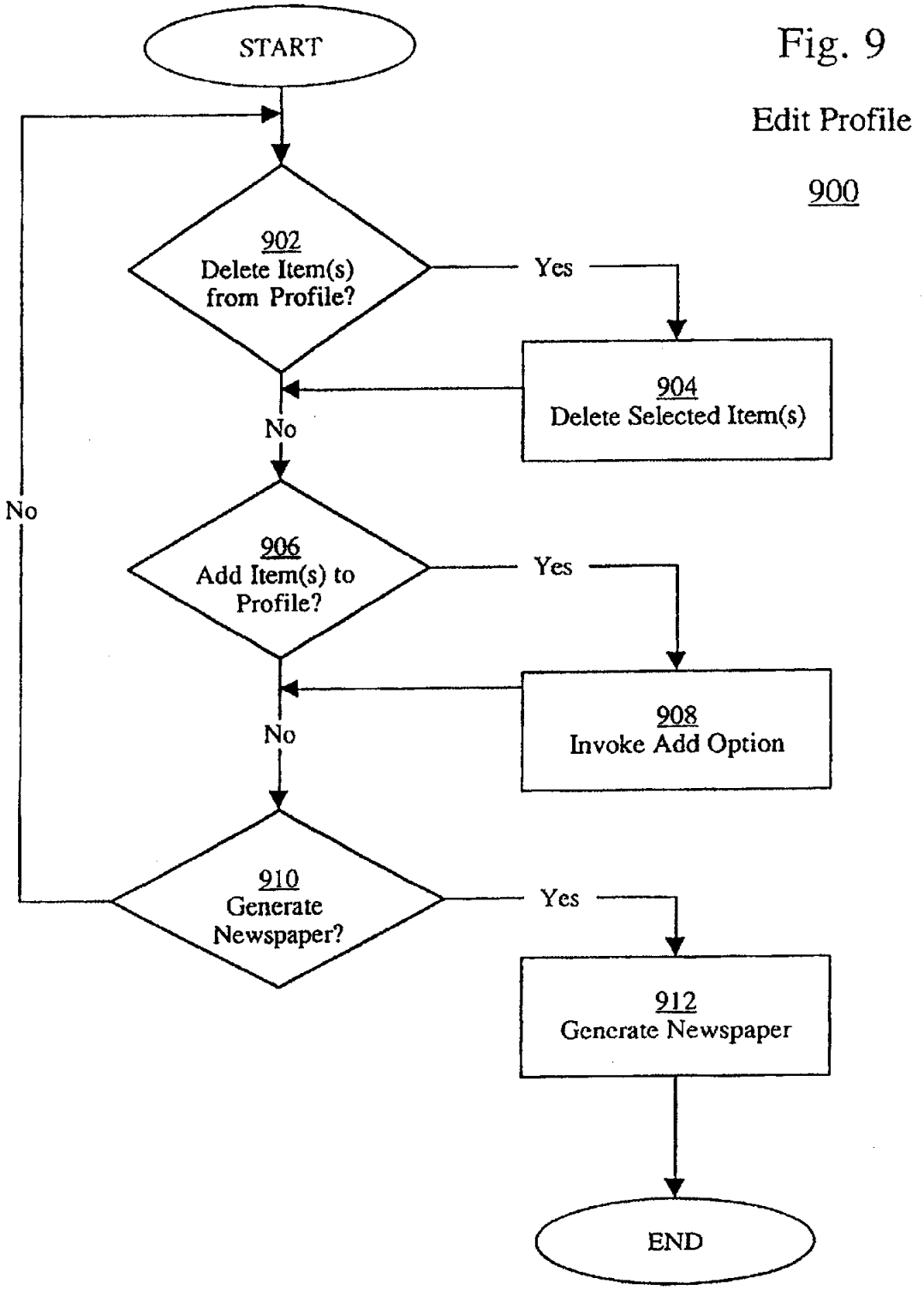

To create or browse an existing profile enter your email address
800
Add to your News Profile
802
Name the topic of interest:
News About Sun
804
Keywords to look for:
Sun Microsystems | SMCC | SunSoft
One of more keywords can be separated by "|".
806
case insensitive search
808
case sensitive search
Search in the following categories: (CTRL – select to select multiple categories)
810a
810b
810c
810d
810
Headline News
US News (Associated Press)
Opinions and Editorials
Businesswire
PR Newswire
Stock Markets
Commodity Markets
Money Markets
Metal Markets
Standard & Poor
Business and Finance News
Agence France Presse
Deutsche Presse-Agentur
Koydo News Agency
812
add to profile

Fig. 8

Fig. 9

Edit Profile

900



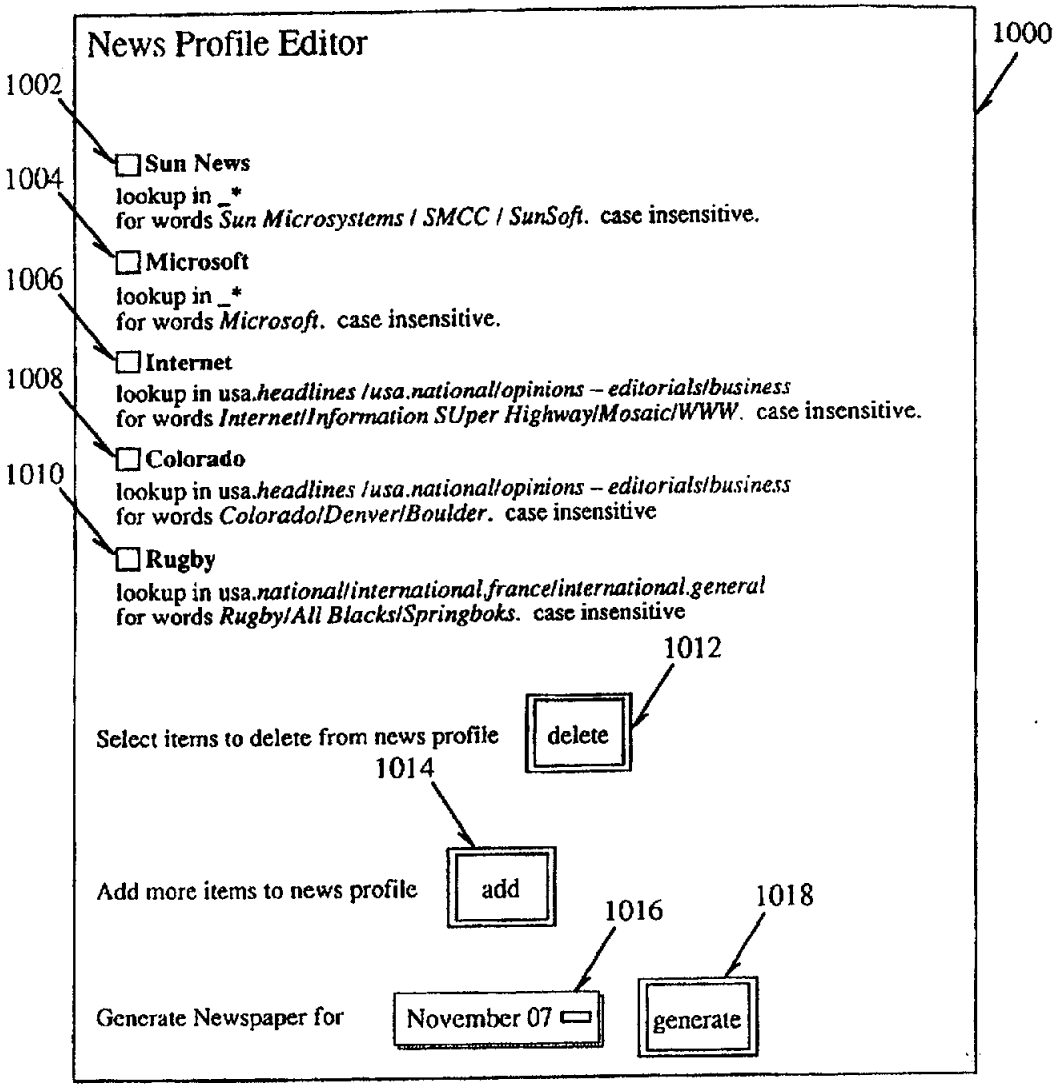

News Profile Editor
1000
1002
Sun News
lookup in _*
for words Sun Microsystems / SMCC / SunSoft. case insensitive.
1004
Microsoft
lookup in _*
for words Microsoft. case insensitive.
1006
Internet
lookup in usa.headlines /usa.national/opinions – editorials/business
for words Internet/Information SUper Highway/Mosaic/WWW. case insensitive.
1008
Colorado
lookup in usa.headlines /usa.national/opinions – editorials/business
for words Colorado/Denver/Boulder. case insensitive
1010
Rugby
lookup in usa.national/international,france/international.general
for words Rugby/All Blacks/Springboks. case insensitive
1012
Select items to delete from news profile
delete
1014
Add more items to news profile
add
1016
1018
Generate Newspaper for
November 07
generate

Fig. 10

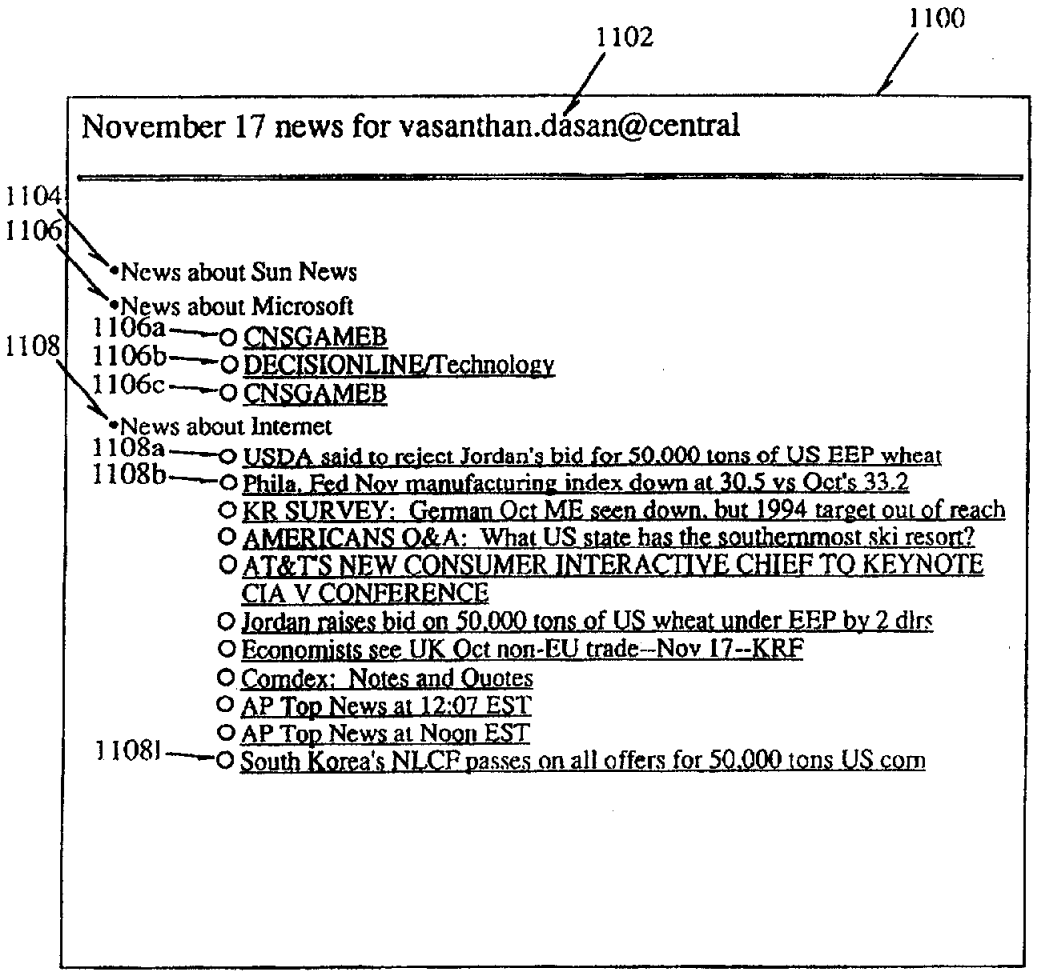

1100
1102
November 17 news for vasanthan.dasan@central
1104
1106
News about Sun News
News about Microsoft
1106a
CNSGAMEB
1108
1106b
DECISIONLINE/Technology
1106c
CNSGAMEB
News about Internet
1108a
USDA said to reject Jordan's bid for 50,000 tons of US EEP wheat
1108b
Phila. Fed Nov manufacturing index down at 30.5 vs Oct's 33.2
KR SURVEY: German Oct ME seen down, but 1994 target out of reach
AMERICANS Q&A: What US state has the southernmost ski resort?
AT&T'S NEW CONSUMER INTERACTIVE CHIEF TO KEYNOTE CIA V CONFERENCE
Jordan raises bid on 50,000 tons of US wheat under EEP by 2 dlrs
Economists see UK Oct non-EU trade--Nov 17--KRF
Comdex: Notes and Quotes
AP Top News at 12:07 EST
AP Top News at Noon EST
1108l
South Korea's NLCF passes on all offers for 50,000 tons US corn

Fig. 11

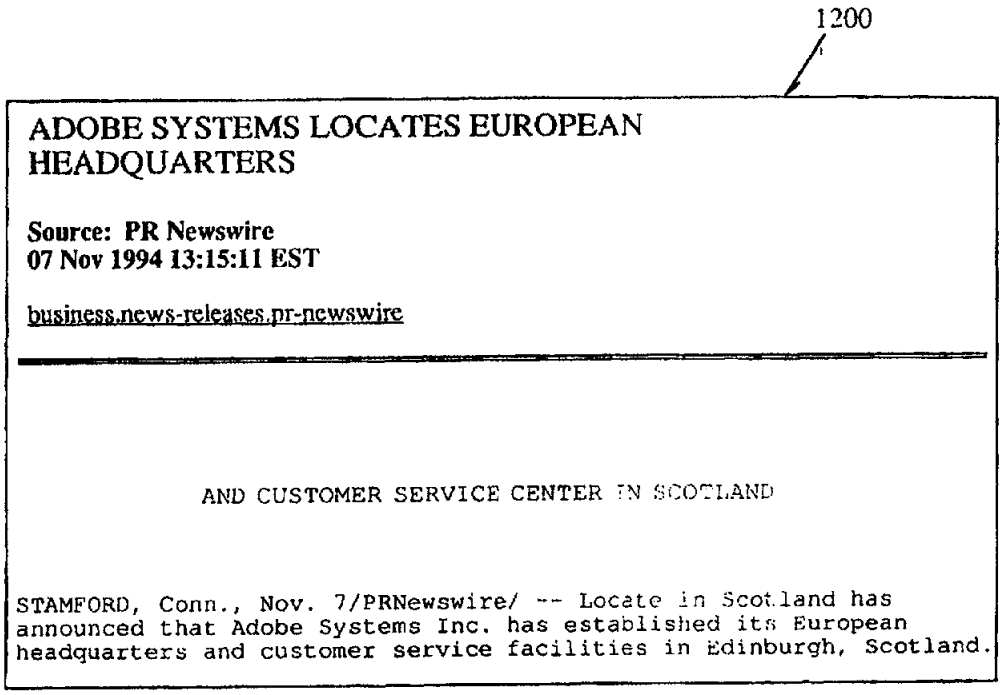

1200
ADOBE SYSTEMS LOCATES EUROPEAN HEADQUARTERS
Source: PR Newswire
07 Nov 1994 13:15:11 EST
business.news-releases.pr-newswire
AND CUSTOMER SERVICE CENTER IN SCOTLAND
STAMFORD, Conn., Nov. 7/PRNewswire/ -- Locate in Scotland has announced that Adobe Systems Inc. has established its European headquarters and customer service facilities in Edinburgh, Scotland.

Fig. 12

PERSONALIZED INFORMATION RETRIEVAL USING USER-DEFINED PROFILE

This is a continuation of application Ser. No. 08/360,997 filed Dec. 20, 1994 and now abandoned.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to information retrieval. More specifically, the present invention relates to a client/server model for information retrieval based upon a user-defined profile, for example, for the generation of an "electronic" newspaper which contains information of interest to a particular user.

2. Background Information

The development of computerized information resources, such as the Internet, and various on-line services, such as Compuserve, America On-line, Prodigy, and other services, has led to a proliferation of electronically-available information. In fact, this electronic information is increasingly displacing more conventional means of information transmission, such as newspapers, magazines, and even, television. The primary issue in all of these resources is filtering the vast amount of information which is available in order that a user obtain that information which is of interest to him.

Currently, a user who subscribes to one of the above-services, or uses the Internet, must manually scan through the various information resources in order to obtain articles, postings, or other files which are of interest. Typically, the user manually scans in areas or sources of interest (publications, USENet Newsgroups, fora, or other areas classified by topic) to find topic areas which may be of interest to the user. The user then retrieves articles or files which have subject headings, for example, matching those which the user wishes to read. For example, certain USENet newsgroups on the Internet have an established etiquette (known as "netiquette"), wherein postings must have relevant subject headings to permit this type of manual scanning. Article headings in newspapers/magazines serve similar functions. Thus, if the user does not wish to view those stories with products for sale, then he simply ignores those postings having the term "For Sale" in the subject heading. Finally, as a last level of filtering, the user can read the stories which have been filtered by topic, and subject heading, and if anytime during the viewing of the story the user wishes to ignore the rest of the article, he can stop reading it and simply discard the story. If desired, the user can download the remaining stories in any number of prior art ways, for off-line viewing.

As is clearly evident, this manual scanning process which a user must engage in is very time-consuming. To some extent, news is already filtered for readers. Those that subscribe to certain magazines/newspapers, or view certain television networks, already have the vast base of electronically-available information (e.g. wire services) filtered for them. However, this "filtering" is performed on a large-scale basis, for a wide audience. The tailoring of specific stories of interest to particular users has not been performed. Thus, a need has arisen to automatically sort through the large variety of electronic sources in order to generate a subset of the stories available in electronic form which is tailored to a user's specific interests is desired.

Thus, the prior art of obtaining user-relevant information from electronic sources suffers from several shortcomings.

SUMMARY

An automatic method and system for retrieving information based on a user-defined profile (e.g. a personalized newspaper). A user-controlled client establishes communication with a stateless server, the server presenting a list of options to the client (e.g. via Hypertext Transfer Protocol (HTTP) exchanges) between the server and the client. The client provides an identification of the user-defined profile. The server engages a first application program (e.g. via a Common Gateway Interface (CGI)), the first application program retrieving the user-defined profile wherein the user-defined profile identifies information which is of interest to the user. The first application program examines a database of information and automatically retrieves a subset of the information from the database based upon which information is of interest to the user as specified in the user-defined profile. The server presents the subset of the information from the database as generated by the first application program to the client.

In one implementation the client comprises an HTTP browser application program and the server comprises an HTTP server application program operative on a remote computer system. The first application program stores a file containing the user-defined profile in order to retain a state of the user-profile, and cause the stateless HTTP server to emulate a server which retains its state from session to session.

In one implementation, the first application program examines the database of information and automatically retrieves a subset of the information from the database based upon the user-defined profile at periodic intervals, in order, for example, to keep the newspaper-up-to-date.

The user-defined profile can include source identifications and associated search terms wherein the first application scans in the information (e.g. a raw news source, USENet newsgroup or other resource) for sources identified by the source identifications. A first set of files in the sources containing the associated search terms may then be identified and the first application program places the first set of files into the subset of the information, for creation of the personalized information—the newspaper. The source identifications and associated search terms from the user-defined profile can be stored by topic wherein the subset is displayed to the user arranged by topic.

The first application program can also cause the server to present options to the user to create or modify the profile, including, a first option to allow the user to specify source identifications in the information and associated search terms to search for in the source identifications to the user-defined profile; and a second option to specify delete and/or change the source identifications and/or the associated search terms in the profile.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying in which like references indicate like elements and in which:

FIG. 1 shows a client/server system in which embodiments of the present invention may be implemented.

FIG. 2 shows a more detailed view of the client/server illustrated in FIG. 1.

FIG. 3 shows the structure of a workstation in which the client or server may be operative.

FIG. 4 shows a more detailed view of the processes operative within the server.

FIGS. *5a* and *5b* shows a sequence of steps performed in the server for generation of a profile and/or creation of a personal newspaper.

FIG. **6** shows the display screen for user control of the process illustrated in FIGS. *5a* and *5b*.

FIG. **7** shows a process of adding a topic to a profile.

FIG. **8** shows a display used for controlling the operation of the process in FIG. **7**.

FIG. **9** shows a process of editing a user profile.

FIG. **10** shows a display used for controlling the editing process.

FIG. **11** shows the results of execution of the personal newspaper generation process according to the user-defined profile.

FIG. **12** shows the viewing of a single article contained in the personal newspaper.

DETAILED DESCRIPTION

The present invention is a method and apparatus for automatically scanning information using a user-defined profile, and providing relevant stories from that information to a user based upon that profile. Although the following will be described with reference to certain particular embodiments, including data structures, flow of steps, hardware configurations, etc. . . , it will be apparent to one skilled in the art that implementations of the present invention can be practiced without these specific details.

Implementations of the present invention use a client/server architecture, as illustrated in FIG. **1**, wherein user requests **110** for news are sent by a client application program **100** to a server **150** (typically, a remote computer system accessible over the Internet or other communication medium). The server **150**, as will be described in more detail below, performs scanning and searching of raw (e.g. unprocessed) information sources (e.g. newswires or newsgroups), based upon these user requests, presents the filtered electronic information as server responses **160** to the client process. The client process may be active in a first computer system, and the server process may be active in a second computer system, and communicate with one another over a communication medium, thus providing distributed functionality and allowing multiple clients to take advantage of the information gathering capabilities of the server.

A more detailed view of the client and server are shown in FIG. **2**. Although the client and server are processes which are operative within two computer systems, these processes being generated from a high level programming language (e.g. C or C++), compiled and executed in a computer system (e.g. a workstation), it can be appreciated by one skilled in the art that they may be implemented in a variety of hardware devices, either programmed or dedicated.

Client **100** and server **150** communicate using the functionally provided by the World-Wide Web (WWW). Clients and servers of the WWW communicate over a communication medium **250** using a standard known as the Hypertext Transfer Protocol (HTTP). In some embodiments, the client and server may be coupled via Serial Line Internet Protocol (SLIP) or TCP/IP connections for high-capacity communication. Active within the client is a first process, known as a "browser" **200**, which establishes the connection with server **150**, and presents information to the user. Any number of commercially or publicly-available browsers may be used, in various implementations, however in this implementation, browser **200** is the Mosaic brand browser (version 2.0 or greater) available from the National Center for Supercomputing Applications (NCSA) in Urbana-Champaign, Ill. Other browsers such as the Netscape, Netcruiser, or the Lynx brand browsers, or others, which are available and provide the functionality specified under HTTP and the Mosaic version **2.0** brand browser or above may be used.

The server **150** executes the corresponding server software which presents information to the client in the form of HTTP responses. The HTTP responses correspond with the web "pages" represented using Hypertext Markup Language (HTML), or other data which is generated by the server, as will now be discussed.

A shortcoming of HTTP is that it is a stateless protocol. The Web "page" from which user transactions may be performed in the server under control of the client is not recalled from client-session to client-session. In this implementation of the present invention the user is able to connect to the remote server and specify a user profile, setting forth his interests. The user is able to specify the context for the information to be searched (e.g. the date). The user is able to save the profile on the remote machine. Finally the user is able to retrieve the personal profile (with any access control, if desired) and edit (add or delete entries) and save it for future operations.

All of this functionality requires an underlying mechanism which is unmet by current versions of HTTP. Under the Mosaic brand browser 2.0 and greater, in addition to HTML functionality **210** provided by the server (display and retrieval of certain textual and other data based upon Hypertext views and selection of item(s)), a Common Gateway Interface (CGI) **220** is provided which allows the client program to direct the server to commence execution of a specified program contained within the server. Using this interface, and HTTP, the server may notify the client of the results of that execution upon completion. The server's application program, the personal newspaper generator, maintains a record of the state of each user's profile, and thus, provides state functionality from session to session to an otherwise stateless protocol.

In order to control the parameters of the execution of this server-resident process, the client may direct the filling out of certain "forms" from his browser. This is also provided by the "fill-in forms" functionality **230** available under Mosaic version 2.0 and greater, which allows the user via his client application program **100**, to specify a "profile" in which the server will cause an application program to function (e.g. the types of stories/articles which are of interest to the user). The details of a user profile will be discussed below.

A computer system, such as a workstation, personal computer or other processing apparatus in which the client **100** or server **150** may be operative is illustrated in FIG. **3**. A workstation in which one implementation of the present invention may be practiced includes system **300**. **300** comprises a bus or other communication means **301** for communicating information, and a processing means **302** coupled with bus **301** for processing information. System **300** further comprises a random access memory (RAM) or other volatile storage device **304** (referred to as main memory), coupled to bus **301** for storing information and instructions to be executed by processor **302**. Main memory **304** also may be used for storing temporary variables or other intermediate information during execution of instructions by processor **302**. System **300** also comprises a read only memory (ROM) and/or other static storage device **306** coupled to bus **301** for storing static information and instruc-

tions for processor **302**, and a data storage device **307** such as a magnetic disk or optical disk and its corresponding disk drive. Data storage device **307** is coupled to bus **301** for storing information and instructions. This may be used for storage of the databases to be described here which maintain information about currently defined problem descriptions using commercially available software products.

System **300** may further be coupled to a display device **321**, such as a cathode ray tube (CRT) or liquid crystal display (LCD) coupled to bus **301** for displaying information to a computer user. Such a display **321** may further be coupled to bus **301** via a frame buffer **310**, which information such as a single or multiple frames or images for display upon display device **321**. An alphanumeric input device **322**, including alphanumeric and other keys, may also be coupled to bus **301** for communicating information and command selections to processor **302**. An additional user input device is cursor control **323**, such as a mouse, a trackball, stylus, or cursor direction keys, coupled to bus **301** for communicating direction information and command selections to processor **302**, and for controlling cursor movement on display **321**.

Note, also, that any or all of the components of system **300** and associated hardware may be used in various embodiments, however, it can be appreciated that any configuration of the system may be used for various purposes according to the particular implementation.

In one embodiment, system **300** is one of the Sun Microsystems® brand family of workstations such as the SPARCstation brand workstation manufactured by Sun Microsystems® of Mountain View, Calif. Processor **302** may be one of the SPARC brand microprocessors manufactured by Sun Microsystems®, Inc. of Mountain View, Calif.

Note that the following discussion of various embodiments discussed herein will refer specifically to a series of routines which are generated in a high-level programming language (e.g., the C or C++ programming language) and compiled, linked, and then run as object code in system **300** during run-time, for example by the SPARCompiler available from SunSoft of Mountain View, Calif. (SPARC and SPARCstation are trademarks of SPARC International, Inc. and are licensed exclusively to Sun Microsystems). These further are used in conjunction with the browser and server software available from NCSA, as described above, including the specification of the appearance of displays in HTML. It can be appreciated by one skilled in the art, however, that the following methods and apparatus may be implemented in special purpose hardware devices, such as discrete logic devices, large scale integrated circuits (LSI's), application-specific integrated circuits (ASIC's), or other specialized hardware. The description here has equal application to apparatus having similar function.

FIG. **4** illustrates the interaction between the server **150** and newspaper generator application program **400**, which is operative under control of the CGI in the server. Depending upon user input, requests are sent to the active application in the server, in this case the personal newspaper generator **400**, which causes the application to perform certain functions. For example, in the case of a profile which does not already exist in the server, newspaper generator **400** allows the creation and editing of search specifications, known as a "profile", on the server. This allows the server to generate the personal newspaper for the client when requested, or at regular intervals, according to implementation.

The user profile or requests which are received from the CGI **220** in the server **150** cause the newspaper generator to perform certain actions. For example, in the case of a profile creation, the profile is created and stored in local files, such as **410**, which may be accessed at a later time by the server during editing operations or during creation of the newspaper. In this way, the stateless protocol of HTTP is transformed into a state-protocol, wherein the client or user profile is recalled from session to session. This also allows periodic automatic generation of the personal newspaper, for example, at regular intervals, such as several times a day or week, according to implementation.

As illustrated, the newspaper generator **400** uses a plurality of raw news sources **420**, **430**, and **440**, which may be any of a number of available raw news feeds. For example, each of these may be a separate news source or wire service (e.g. Reuters, Associated Press), while another may be an electronic discussion group (e.g. a USENet newsgroup). Any number or types of sources may be used, according to implementation.

FIGS. **5***a* and **5***b* illustrates the logic of the operation of the personal newspaper generator. First, at step **502**, the newspaper's front page, or welcome screen is displayed. This is the page which is displayed when the use first accesses the server via the specification of a URL (Uniform Resource Locator). A display screen, such as **600** illustrated in FIG. **6**, is displayed upon the user's console. Options and other data entered on the form displayed on the console are committed, causing invocation of the newspaper generation program, upon selection by the user of the "doit" icon **610** on display **600**. At step **504**, the user can then enter, via the forms support in the browser, a profile or user name in field **602**. This is enabled by selecting using a selection device or other means, and filling in the selected blank with the profile name (typically, an e-mail address of the user). If desired in the given implementation, access control may be performed at step **506**, wherein the user is queried for a password.

If the user wishes to edit the profile as detected at step **508**, which is indicated by the selection of the user interface object **604**, then process **500** proceeds to FIG. **5***b*. If not (user interface object **606** is selected), then the user may specify a date for which the personal newspaper will be generated, such as by filling in the field **608** in the user interface form, by a pull-down, pop-up menu, or other means. Then, at step **512** it is determined whether the profile for the user exists. If so, then the newspaper is generated using the stored profile. According to the topics/subject/keywords selected by the user in his profile, the raw news source(s) are scanned, and that information matching the profile is gathered at step **514**. If the profile does not exist, then at step **516**, the user is given the options for creating and editing the profile.

If the user desired to edit and/or lookup the profile (icon **604** is selected) then, it is determined at step **518** on FIG. **5***b* whether the profile exists. If not, then the user is given the option to create a new profile. Then, the user can edit the profile, if created or it already exists, at step **522**. Upon completion of any editing operations, the process is complete and returns to FIG. **5***b*.

A personal news profile is stored in a ASCII text file by topic by the newspaper generator. Each topic contains a number of sources (e.g. publications or newsfeeds) to search and corresponding search term(s). As in certain full-text retrieval systems, search terms may be delimited by disjunctions (the "|" character). An example of a news profile is as follows:

```
Sun News~.*~Sun MicrosystemslSMCCISunSoft~insensitive
Microsoft~.*~Microsoft~insensitive
Internet~usa.headlinesIusa.nationalIopinions-editorials|
businesswirelprnewswirelbusiness.financel
international.francelinternational.germanylinternational.japan
international.other~InternetlInformation
SuperHighwaylMosaicIWWW~insensitive
Colorado~usa.headlinesIusa.nationalIopinionseditorials|
businesswirelpr-newswirelbusiness.financelinternational.france
international.germanylinternational.japanlinternational.other~
ColoradolDenverlBoulder~insensitive
```

Each of the lists of strings for topics, sources and keywords are delimited in the file by the "~" character. The initial string (e.g. "Sun News") identifies the topic, the second set of strings (e.g. ".*") identifies the files specifications for the sources (wherein ".*" indicates searching in all raw sources), the next set of strings are the search terms, and the final string indicates whether the search terms are case sensitive or not. The creation and editing of a profile will now be discussed.

The adding of topics to a news profile is shown in FIGS. **7** and **8**. FIG. **7** shows the process steps performed by the profile generator portion of the newspaper generator, and FIG. **8** shows the user interface which controls the adding of topics to the profile. First, at step **702**, the user labels the topic of interest, such as by filling in the field **802** provided by the fill-in forms feature of HTML or CGI. At step **704**, the user specifies search terms used in the full-text search. These are illustrated in field **804**. Any number of search terms may be used and the "|" character is treated as a disjunction ("or"). Then, by selecting either of user interface objects **806** or **808**, the user specifies whether the search terms are case sensitive or not. This is detected at step **706**. At step **708**, using either a scrollable list containing selectable item(s), as illustrated in field **810**, or other means, the user specifies the search context(s) (the publications, newsfeeds, etc. . .) in which to search. By the selection of icon **812** or other commit means, the data entered into the screen **800** at step **710** is added to the user profile for generation of the newspaper.

The process steps and user interface for editing a user profile is shown in FIGS. **9** and **10**. For example, for a particular profile as displayed on screen **1000**, any of the selection icons **1002–1010** may be selected for modification of that topic. At step **902** it is determined whether the user has selected the "delete" option for deleting topics from the profile (e.g. the selection of icon **1012** on screen **1000**). If so, then the selected item(s) are deleted at step **904**. Then it is determined at step **906** whether the user has selected the "add" icon **1014**. If so, then the add option (process **700** of FIG. **7**) is invoked at step **908**. It is then determined at step **910** whether the generation of the newspaper based upon the profile is requested by the user. This is indicated by the user selecting icon **1018** (including, specifying a date for which the newspaper should be generated in field **1016**). The specification of the date limits the scope of the search to the date(s) specified. If generation of the newspaper is requested, then the newspaper is generated based upon the profile at step **912**.

As previously discussed, the process for actually creating the newspaper functions as a full-text retrieval system which is controlled by the context(s) (e.g. source(s) and date(s)) and corresponding search term(s) specified in each of the topics contained in the profile. This functions much in the same way as other on-line full-text retrieval systems except that it may be automatically performed by the server on demand or at periodic intervals, and that it is under control of the stored profile. The mechanics of this will be briefly discussed.

In implemented embodiments, raw news sources come in as large text files with a predetermined format. The full-text searching process performed during newspaper generation is executed upon these large files. Upon location of certain search terms having the specified context(s) in the file, the portion of the file containing the located article is parsed so as to be converted into an individual ASCII file which is resident on the server. The file names of the articles are unique, for unique reference using HTML. In implemented embodiments, the article file names are in numeric form and are the year, month, date, hour, minute and second the article was created all concatenated in order to be unique (no two articles are received exactly the same second). Then, via a second parsing process, the title of the article is used to create an anchor in the HTML page representation of the personal newspaper, which references the article by its file specification. Upon selection by the user, the text of the article can then be referenced at the server, and displayed as a second HTML page.

FIG. **11** shows the results of the creation of a personal newspaper—the personal newspaper main screen. This is displayed at the client as a result of the full-text searching, the parsing and HTML page generation process performed at the server as discussed above. As illustrated in screen **1100**, the profile name is displayed as **1102**. Each of the topics from the profile is listed in a first level heading, such as **1104**, **1106**, **1108**, etc. . . Each of the first level topic headings are followed by second level headings, such as **1106***a*–**1106***c*, **1108***a*–**1108***l*, etc. . . . listing the names of stories, if any, which match the search terms and sources. Via the selection by the user of the headings for each of the stories (the above-mentioned anchors), the text for the story may be referenced and viewed. An example of the display of an article from the personal newspaper is illustrated in screen **1200** of FIG. **12**. Via HTML, a reference may also be inserted in the HTML page containing the article in order to allow the user to return to the personal newspaper main screen (not shown).

Thus, method and apparatus for a client application program retrieving data from a server based upon a defined and stored user profile of desired information has been described. Note that though the foregoing has particular utility and has been described with reference to certain specific embodiments in the figures and the text, that one may practice the present invention without implementing all of these specific details. Thus, the figures and the text are to be viewed an illustrative sense only, and not limit the present invention. The present invention is only to be limited by the appended claims which follow.

What is claimed is:

1. A computer-implemented method of retrieving information comprising the following steps:

a. a user-controlled client establishing communication with a server over a communication channel;

b. said client identifying a user-defined profile to said server;

c. said server engaging a first application program, said first application program retrieving said user-defined profile;

d. said first application program examining a database of information and automatically retrieving a subset of said information from said database based upon said user-defined profile; and

e. said first application program transmitting said subset of said information from said database to said server

and said server presenting said subset of said information to said client.

2. The method of claim 1 wherein said client comprises an HTTP (Hypertext Transfer Protocol) browser active on a first computer system and said server comprises an HTTP server application program active on a second computer system.

3. The method of claim 2 wherein said HTTP server application program communicates with said first application program active on said second computer system via a Common Gateway Interface (CGI).

4. The method of claim 3 further comprising the step of said first application program storing a file in said second computer system containing said user-defined profile in order to retain a state of said user-profile.

5. The method of claim 2 wherein said step of said first application program examining said database of information and automatically retrieving a subset of said information from said database based upon said user-defined profile is performed at periodic intervals irrespective of said client establishing communication with said server.

6. The method of claim 1 wherein said step of said first application program examining said database of information and automatically retrieving a subset of said information from said database based upon said user-defined profile includes:

a. said first application program retrieving source identifications and associated search terms from said user-defined profile;

b. said first application scanning in said database of information for sources identified by said source identifications and identifying a first set of files in said sources containing said associated search terms; and

c. said first application program placing said first set of files into said subset of said information.

7. The method of claim 1 further comprising the step of said first application program causing said server to present options to said user to create or modify said profile, said options including:

a. a first option allowing said user to specify source identifications in said information and associated search terms to search for in said source identifications to said user-defined profile; and

b. a second option allowing said user to specify delete and/or change said source identifications and/or said associated search terms in said profile.

8. An automatic method of retrieving information comprising the following steps:

a. a client under control of a user establishing communication with a server via HTTP Hypertext Transfer Protocol) exchanges between said client and said server;

b. said client providing an identification of a user-defined profile to said server;

c. said server engaging a first application program, said first application program retrieving said user-defined profile wherein said user-defined profile identifies information which is of interest to said user;

d. said first application program examining a database of information and automatically retrieving a subset of said information from said database based upon which information is of interest to said user as specified in said user-defined profile; and

e. said server presenting said subset of said information from said database to said client as retrieved by said first application program.

9. The method of claim 8 wherein said client comprises an HTTP (Hypertext Transfer Protocol) browser application program active on a first computer system and said server comprises an HTTP server application program operative on a remote computer system.

10. The method of claim 9 further comprising the step of said first application program storing a file containing said user-defined profile in said remote computer system in order to retain a state of said user-profile.

11. The method of claim 8 wherein said step of said first application program examining said database of information and automatically retrieving a subset of said information from said database based upon said user-defined profile is performed at periodic intervals irrespective of said client establishing communication with said server.

12. The method of claim 8 wherein said step of said first application program examining said database of information and automatically retrieving a subset of said information from said database based upon said user-defined profile includes:

a. said first application program retrieving source identifications and associated search terms from said user-defined profile;

b. said first application scanning in said database of information for sources identified by said source identifications and identifying a first set of files in said sources containing said associated search terms; and

c. said first application program placing said first set of files into said subset of said information.

13. The method of claim 12 wherein said source identifications and associated search terms from said user-defined profile are stored by topic.

14. The method of claim 8 further comprising the step of said first application program causing said server to present options to said user to create or modify said profile, said options including:

a. a first option allowing said user to specify source identifications in said information and associated search terms to search for in said source identifications to said user-defined profile; and

b. a second option allowing said user to specify delete and/or change said source identifications and/or said associated search terms in said profile.

15. The method of claim 14 wherein said source identifications and said associated search terms are organized by topics.

16. The method of claim 15 wherein said server presenting said subset of said information from said database as generated by said first application program comprises presenting said subset organized by said topics.

17. A system for retrieving information comprising:

a. a server having an interface for engaging a first application program responsive to a client providing an identification of a user-defined profile;

b. said server having an execution control interface for causing said first application program to examine a database of information and automatically retrieve a subset of said information based upon which information is of interest to said user as identified in said user-defined profile; and

c. said server having a presentation circuit for presenting said subset of said information from said database to said client responsive to a request by said client.

18. The system of claim 17 wherein said client comprises an HTTP (Hypertext Transfer Protocol) browser operative on a first computer system and said server comprises an HTTP server operative on a remote computer system.

19. The system of claim 17 wherein said server further includes a profile circuit for allowing said client to specify said user-defined profile and storing a file containing said user-defined profile in order to retain a state of said user-defined profile.

20. The system of claim 17 wherein said execution control interface comprises an activation circuit for activating said first application program at periodic intervals to cause said first application program to perform said examining said database of information and automatically retrieving a subset of said information from said database based upon said user-defined profile.

21. The system of claim 17 wherein said first application program examining said database of information and automatically retrieving a subset of said information from said database based upon said user-defined profile includes:

a. said first application program retrieving source identifications and associated search terms from said user-defined profile;

b. said first application scanning in said database of information for sources identified by said source identifications and identifying a first set of files in said sources containing said associated search terms; and

c. said first application program placing said first set of files into said subset of said information.

22. The system of claim 21 wherein said source identifications and associated search terms from said user-defined profile are stored by topic.

23. The system of claim 17 further comprising said server having an option presentation circuit for presenting said options to said user to create or modify said profile, said options including:

a. a first option allowing said user to specify source identifications in said information and associated search terms to search for in said source identifications of said user-defined profile; and

b. a second option allowing said user to specify delete and/or change said source identifications and/or said associated search terms in said profile.

24. The system of claim 23 wherein said source identifications and said associated search terms are organized by topics.

25. The system of claim 24 wherein said presentation circuit for presenting said subset of said information from said database as generated by said first application program comprises a topic presentation circuit for presenting said subset of said information organized by said topics.

26. A computer-implemented method of retrieving information comprising the following steps:

a. a user-controlled client establishing communication with a server;

b. said client identifying a user-defined profile to said server;

c. said server engaging a first application program, said first application program retrieving said user-defined profile;

d. said first application program examining a database of information and automatically retrieving a subset of said information from said database based upon said user-defined profile; and

e. said first application program transmitting said subset of said information from said database to said server and said server presenting said subset of said information to said client.

* * * * *

CIVIL COVER SHEET

The JS 44 civil cover sheet and the information contained herein neither replace nor supplement the filing and service of pleadings or other papers as required by law, except as provided by local rules of court. This form, approved by the Judicial Conference of the United States in September 1974, is required for the use of the Clerk of Court for the purpose of initiating the civil docket sheet. (SEE INSTRUCTIONS ON THE REVERSE OF THE FORM.)

I. (a) PLAINTIFFS
SUN MICROSYSTEMS, INC.

DEFENDANTS
VERSATA ENTERPRISES, INC.,
(see full list of defendants on attached sheet)

(b) County of Residence of First Listed Plaintiff _____
(EXCEPT IN U.S. PLAINTIFF CASES)

County of Residence of First Listed Kent
(IN U.S. PLAINTIFF CASES ONLY)
NOTE: IN LAND CONDEMNATION CASES, USE THE LOCATION OF THE LAND INVOLVED.

(c) Attorney's (Firm Name, Address, and Telephone Number)

Skadden, Arps, Slate, Meagher & Flom LLP
Paul J. Lockwood
One Rodney Square
Wilmington, DE 19899
Tel: (302) 651-3000

Jeffrey G. Randall, David W. Hansen
525 University Avenue, Suite 1100
Palo Alto, CA 94301
(650) 470-4500

Attorneys (If Known)

II. BASIS OF JURISDICTION (Place an "X" in One Box Only)

- ☐ 1 U.S. Government Plaintiff
- ☒ 3 Federal Question (U.S. Government Not a Party)
- ☐ 2 U.S. Government Defendant
- ☐ 4 Diversity (Indicate Citizenship of Parties in Item III)

III. CITIZENSHIP OF PRINCIPAL PARTIES (Place an "X" in One Box for Plaintiff and One Box for Defendant)
(For Diversity Cases Only)

	PTF	DEF		PTF	DEF
Citizen of This State	☐ 1	☐ 1	Incorporated *or* Principal Place of Business In This State	☒ 4	☒ 4
Citizen of Another State	☐ 2	☐ 2	Incorporated *and* Principal Place of Business In Another State	☐ 5	☐ 5
Citizen or Subject of a Foreign Country	☐ 3	☐ 3	Foreign Nation	☐ 6	☐ 6

IV. NATURE OF SUIT (Place an "X" in One Box Only)

CONTRACT
- ☐ 110 Insurance
- ☐ 120 Marine
- ☐ 130 Miller Act
- ☐ 140 Negotiable Instrument
- ☐ 150 Recovery of Overpayment & Enforcement of Judgment
- ☐ 151 Medicare Act
- ☐ 152 Recovery of Defaulted Student Loans (Excl. Veterans)
- ☐ 153 Recovery of Overpayment of Veteran's Benefits
- ☐ 160 Stockholders' Suits
- ☐ 190 Other Contract
- ☐ 195 Contract Product Liability
- ☐ 196 Franchise

REAL PROPERTY
- ☐ 210 Land Condemnation
- ☐ 220 Foreclosure
- ☐ 230 Rent Lease & Ejectment
- ☐ 240 Torts to Land
- ☐ 245 Tort Product Liability
- ☐ 290 All Other Real Property

TORTS

PERSONAL INJURY
- ☐ 310 Airplane
- ☐ 315 Airplane Product Liability
- ☐ 320 Assault, Libel & Slander
- ☐ 330 Federal Employers' Liability
- ☐ 340 Marine
- ☐ 345 Marine Product Liability
- ☐ 350 Motor Vehicle
- ☐ 355 Motor Vehicle Product Liability
- ☐ 360 Other Personal Injury

PERSONAL INJURY
- ☐ 362 Personal Injury—Med. Malpractice
- ☐ 365 Personal Injury — Product Liability
- ☐ 368 Asbestos Personal Injury Product Liability

PERSONAL PROPERTY
- ☐ 370 Other Fraud
- ☐ 371 Truth in Lending
- ☐ 380 Other Personal Property Damage
- ☐ 385 Property Damage Product Liability

CIVIL RIGHTS
- ☐ 441 Voting
- ☐ 442 Employment
- ☐ 443 Housing/ Accommodations
- ☐ 444 Welfare
- ☐ 445 Amer. w/Disabilities - Employment
- ☐ 446 Amer. w/Disabilities – Other
- ☐ 440 Other Civil Rights

PRISONER PETITIONS
- ☐ 510 Motions to Vacate Sentence

Habeas Corpus:
- ☐ 530 General
- ☐ 535 Death Penalty
- ☐ 540 Mandamus & Other
- ☐ 550 Civil Rights
- ☐ 555 Prison Condition

FORFEITURE/PENALTY
- ☐ 610 Agriculture
- ☐ 620 Other Food & Drug
- ☐ 625 Drug Related Seizure of Property 21 USC 881
- ☐ 630 Liquor Laws
- ☐ 640 R.R. & Truck
- ☐ 650 Airline Regs.
- ☐ 660 Occupational Safety/Health
- ☐ 690 Other

LABOR
- ☐ 710 Fair Labor Standards Act
- ☐ 720 Labor/Mgmt. Relations
- ☐ 730 Labor/Mgmt.Reporting & Disclosure Act
- ☐ 740 Railway Labor Act
- ☐ 790 Other Labor Litigation
- ☐ 791 Empl. Ret. Inc. Security Act

BANKRUPTCY
- ☐ 422 Appeal 28 USC 158
- ☐ 423 Withdrawal 28 USC 157

PROPERTY RIGHTS
- ☐ 820 Copyrights
- ☒ 830 Patent
- ☐ 840 Trademark

SOCIAL SECURITY
- ☐ 861 HIA (1395ff)
- ☐ 862 Black Lung (923)
- ☐ 863 DIWC/DIWW (405(g))
- ☐ 864 SSID Title XVI
- ☐ 865 RSI (405(g))

FEDERAL TAX SUITS
- ☐ 870 Taxes (U.S. Plaintiff or Defendant)
- ☐ 871 IRS—Third Party 26 USC 7609

OTHER STATUTES
- ☐ 400 State Reapportionment
- ☐ 410 Antitrust
- ☐ 430 Banks and Banking
- ☐ 450 Commerce
- ☐ 460 Deportation
- ☐ 470 Racketeer Influenced and Corrupt Organizations
- ☐ 480 Consumer Credit
- ☐ 490 Cable/Sat TV
- ☐ 810 Selective Service
- ☐ 850 Securities/Commodities/ Exchange
- ☐ 875 Customer Challenge 12 USC 3410
- ☐ 890 Other Statutory Actions
- ☐ 891 Agricultural Acts
- ☐ 892 Economic Stabilization Act
- ☐ 893 Environmental Matters
- ☐ 894 Energy Allocation Act
- ☐ 895 Freedom of Information Act
- ☐ 900Appeal of Fee Determination Under Equal Access to Justice
- ☐ 950 Constitutionality of State Statutes

V. ORIGIN (Place an "X" in One Box Only)

☒ 1 Original Proceeding | ☐ 2 Removed from State Court | ☐ 3 Remanded from Appellate Court | ☐ 4 Reinstated or Reopened | ☐ 5 Transferred from another district (specify) | ☐ 6 Multidistrict Litigation | ☐ 7 Appeal to District Judge from Magistrate Judgment

VI. CAUSE OF ACTION

Cite the U.S. Civil Statute under which you are filing **(Do not cite jurisdictional statutes unless diversity):**
35 U.S.C. Section 1, et seq.

Brief description of cause:
Patent Infringement

VII. REQUESTED IN COMPLAINT: ☐ CHECK IF THIS IS A **CLASS ACTION** UNDER F.R.C.P. 23 **DEMAND $**

CHECK YES only if demanded in complaint:
JURY DEMAND: ☒ Yes ☐ No

VIII. RELATED CASE(S) IF ANY (See instructions): JUDGE _____ DOCKET NUMBER _____

DATE
November 30, 2007

SIGNATURE OF ATTORNEY OF RECORD

FOR OFFICE USE ONLY

RECEIPT # _____ AMOUNT _____ APPLYING IFP _____ JUDGE _____ MAG. JUDGE _____

American LegalNet, Inc. www.USCourtForms.com

Attachment to Civil Cover Sheet

SUN MICROSYSTEMS, INC.

Plaintiff,

vs.

VERSATA ENTERPRISES, INC.,
VERSATA SOFTWARE, INC.,
VERSATA DEVELOPMENT GROUP, INC.,
VERSATA COMPUTER INDUSTRY SOLUTIONS, INC., |
VERSATA, INC.,
TRILOGY, INC., and
NEXTANCE, INC. and

Defendants.

AO FORM 85 RECEIPT (REV. 9/04)

United States District Court for the District of Delaware

Civil Action No. 07-782-

ACKNOWLEDGMENT OF RECEIPT FOR AO FORM 85

NOTICE OF AVAILABILITY OF A UNITED STATES MAGISTRATE JUDGE TO EXERCISE JURISDICTION

I HEREBY ACKNOWLEDGE RECEIPT OF 2 COPIES OF AO FORM 85.

11/30/07
(Date forms issued)

x Allan Nicholson 11-30-2007
(Signature of Party or their Representative)

x Allan Nicholson 11-30-2007
(Printed name of Party or their Representative)

Note: Completed receipt will be filed in the Civil Action