US006615253B1

(12) **United States Patent**
Bowman-Amuah

(10) **Patent No.: US 6,615,253 B1**
(45) **Date of Patent: Sep. 2, 2003**

(54) **EFFICIENT SERVER SIDE DATA RETRIEVAL FOR EXECUTION OF CLIENT SIDE APPLICATIONS**

(75) Inventor: **Michel K. Bowman-Amuah**, Colorado Springs, CO (US)

(73) Assignee: **Accenture LLP**, Palo Alto, 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.: **09/387,430**

(22) Filed: **Aug. 31, 1999**

(51) **Int. Cl.**[7] **G06F 15/16**; G06F 12/00; G06F 17/00

(52) **U.S. Cl.** **709/219**; 711/118; 707/100

(58) **Field of Search** 709/217, 218, 709/219, 203, 234, 231, 232; 707/100; 711/118

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,047,918	A	9/1991	Schwartz et al.	707/203
5,133,075	A	7/1992	Risch	707/201
5,187,787	A	2/1993	Skeen et al.	709/314
5,241,580	A	8/1993	Babson, III	379/15
5,291,593	A	3/1994	Abraham et al.	707/103
5,301,270	A	4/1994	Steinberg et al.	345/326
5,301,320	A	4/1994	McAttee et al.	395/650
5,313,636	A	5/1994	Noble et al.	707/1
5,414,812	A	5/1995	Filip et al.	707/103
5,434,978	A	7/1995	Dockter et al.	709/230
5,437,038	A	7/1995	Silberbauer et al.	395/700
5,457,797	A	10/1995	Butterworth et al.	709/302
5,463,686	A	10/1995	Lebourges	379/220
5,471,629	A	11/1995	Risch	707/201
5,475,844	A	12/1995	Shiramizu et al.	709/104
5,499,371	A	3/1996	Henninger et al.	717/2
5,560,005	A	9/1996	Hoover et al.	707/10
5,568,644	A	10/1996	Nelson et al.	395/741
5,581,758	A	12/1996	Burnett et al.	707/103
5,606,664	A	2/1997	Brown et al.	709/224
5,613,155	A *	3/1997	Baldiga et al.	709/232
5,623,418	A	4/1997	Rostoker et al.	716/1
5,642,511	A	6/1997	Chow et al.	395/701
5,649,139	A	7/1997	Weinreb et al.	707/202
5,671,386	A	9/1997	Blair et al.	395/405

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	0123456 A2	1/2000	100/100
WO	WO92/01251	1/1992	

(List continued on next page.)

OTHER PUBLICATIONS

Kovalerchuck et al., comparison of relational methods and attribute–based methods for data mining in intelligent systems, proceedings of the 1999 IEEE, International Symposium on Intelligent Systems and Semiotics, Cambridge, MA, pp 162–166. Date Sep. 1999.

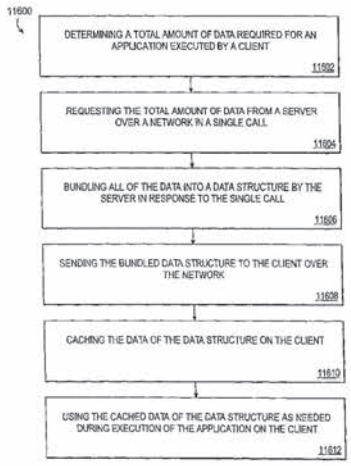
(List continued on next page.)

Primary Examiner—Krisna Lim
(74) *Attorney, Agent, or Firm*—Oppenheimer Wolff & Donnelly LLP

(57) **ABSTRACT**

A system, method, and article of manufacture are provided for efficiently retrieving data. A total amount of data required for an application executed by a client is determined. In a single call, the total amount of data from a server is requested over a network. All of the data is bundled into a data structure by the server in response to the single call. The bundled data structure is sent to the client over the network and the data of the data structure is cached on the client. The cached data of the data structure is used as needed during execution of the application on the client.

18 Claims, 123 Drawing Sheets



U.S. PATENT DOCUMENTS

5,675,748	A		10/1997	Ross	395/284
5,677,997	A		10/1997	Talatik	706/45
5,680,602	A		10/1997	Bloem et al.	707/1
5,692,107	A		11/1997	Simoudis et al.	706/12
5,706,506	A		1/1998	Jensen et al.	707/103
5,708,828	A		1/1998	Coleman	395/785
5,710,901	A		1/1998	Srodghill et al.	345/339
5,715,397	A		2/1998	Ogawa et al.	395/200.18
5,721,908	A		2/1998	Lagarde et al.	395/610
5,724,575	A		3/1998	Hoover et al.	707/10
5,732,218	A	*	3/1998	Bland et al.	709/224
5,732,263	A		3/1998	Havens et al.	707/103
5,732,270	A		3/1998	Foody et al.	709/303
5,737,607	A		4/1998	Hamilton et al.	395/701
5,751,965	A		5/1998	Mayo et al.	709/224
5,758,351	A		5/1998	Gibson et al.	707/104
5,761,513	A		6/1998	Yellin et al.	395/705
5,764,235	A	*	6/1998	Hunt et al.	345/428
5,764,955	A		6/1998	Doolan	709/223
5,774,660	A		6/1998	Brendel et al.	709/201
5,778,368	A		7/1998	Hogan et al.	707/10
5,787,413	A		7/1998	Kauffman et al.	707/2
5,799,310	A		8/1998	Anderson et al.	707/102
5,867,153	A		2/1999	Grandcolas et al.	345/326
5,870,742	A		2/1999	Chang et al.	707/8
5,870,746	A		2/1999	Knutson et al.	707/101
5,872,973	A		2/1999	Mitchell et al.	709/332
5,873,086	A		2/1999	Fujii et al.	707/10
5,878,408	A		3/1999	Van Huben et al.	707/1
5,890,133	A		3/1999	Ernst	705/7
5,892,909	A		4/1999	Grasso et al.	709/201
5,896,383	A		4/1999	Wakeland	370/400
5,898,870	A		4/1999	Okuda et al.	709/104
5,905,873	A		5/1999	Hartmann et al.	395/200.79
5,905,897	A		5/1999	Chan et al.	395/733
5,907,704	A		5/1999	Gudmundson et al.	395/701
5,909,540	A		6/1999	Carter et al.	714/4
5,915,115	A		6/1999	Talati	717/5
5,918,004	A	*	6/1999	Anderson et al.	714/25
5,920,703	A		7/1999	Campbell et al.	395/200.66
5,933,816	A		8/1999	Zeannah et al.	705/35
5,940,075	A		8/1999	Mutschler, III et al.	345/335
5,940,594	A	*	8/1999	Ali et al.	709/203
5,946,694	A		8/1999	Copeland et al.	707/103
5,946,697	A		8/1999	Shen	707/104
5,953,707	A		9/1999	Huang et al.	705/10
5,958,012	A		9/1999	Battat et al.	709/224
5,960,200	A		9/1999	Eager et al.	717/5
5,966,451	A		10/1999	Utsumi	380/49
5,987,247	A		11/1999	Lau	717/2
5,987,501	A	*	11/1999	Hamilton et al.	709/203
5,987,514	A		11/1999	Rangarajan	709/224
5,987,633	A		11/1999	Newman et al.	714/712
5,995,753	A		11/1999	Walker	717/2
5,995,945	A		11/1999	Notani et al.	705/28
5,999,948	A		12/1999	Nelson	
5,999,972	A	*	12/1999	Gish	709/203
6,006,230	A		12/1999	Ludwig et al.	707/10
6,016,394	A		1/2000	Walker	717/1
6,018,743	A		1/2000	Xu	707/103 R
6,023,722	A		2/2000	Colyer	709/201
6,029,174	A		2/2000	Sprenger et al.	707/103
6,029,177	A		2/2000	Sadiq et al.	707/201
6,029,196	A	*	2/2000	Lenz	709/203
6,032,153	A		2/2000	Sadiq et al.	707/103
6,035,303	A		3/2000	Baer et al.	707/103
6,038,598	A	*	3/2000	Danneels	709/219
6,041,365	A		3/2000	Kleinerman	709/302
6,047,357	A	*	4/2000	Bannon et al.	711/122
6,052,739	A		4/2000	Bopardikar et al.	709/301
6,057,856	A		5/2000	Miyashita et al.	345/435
6,070,191	A		5/2000	Narendran et al.	709/226
6,078,960	A		6/2000	Ballard	709/229
6,081,837	A		6/2000	Stedman et al.	709/219
6,083,276	A		7/2000	Davidson et al.	717/1
6,085,198	A		7/2000	Skinner et al.	707/103
6,092,118	A		7/2000	Tsang	709/246
6,108,703	A		8/2000	Leighton et al.	709/226
6,115,752	A		9/2000	Chauhan	709/241
6,125,359	A		9/2000	Lautzenheiser et al.	706/60
6,128,279	A		10/2000	O'Neil et al.	370/229
6,141,660	A		10/2000	Bach et al.	345/352
6,141,759	A	*	10/2000	Braddy	713/201
6,144,991	A		11/2000	England	709/205
6,148,335	A		11/2000	Haggard et al.	709/224
6,148,361	A		11/2000	Carpenter et al.	710/260
6,154,212	A		11/2000	Eick et al.	345/356
6,157,940	A		12/2000	Marullo et al.	709/22
6,182,182	B1		1/2001	Bradley et al.	710/129
6,202,099	B1		3/2001	Gillies et al.	709/317
6,223,209	B1		4/2001	Watson	709/201
6,243,392	B1	*	6/2001	Uemura et al.	370/465
6,243,761	B1	*	6/2001	Mogul et al.	709/246
6,272,556	B1	*	8/2001	Gish	709/315
6,321,274	B1	*	11/2001	Shakib et al.	709/328

FOREIGN PATENT DOCUMENTS

WO	WO 99/08208	2/1999	G06F/17/30
WO	WO 99/44155	9/1999	
WO	PCT/US00/23885	8/2000	
WO	PCT/US00/23999	8/2000	
WO	PCT/US00/24082	8/2000	
WO	PCT/US00/24083	8/2000	
WO	PCT/US00/24084	8/2000	
WO	PCT/US00/24085	8/2000	
WO	PCT/US00/24086	8/2000	
WO	PCT/US00/24125	8/2000	
WO	PCT/US/00/24188	8/2000	
WO	PCT/US00/24189	8/2000	
WO	PCT/US00/24236	8/2000	

OTHER PUBLICATIONS

Kinexis. Object–orientation and Transaction Processing Where Do They Meet. OOPSLA Keynote, Oct. 6–11, 1991.

Lee et al. Path Dictionary: A New Access Method for Query Processing in Object–oriented Databases. IEEE Transactions on Knowledge and Data Engineering, v10, n3, May/Jun. 1998.

Buddrus et al. Enacting Authorization Models for Object–oriented Databases. Database and Expert Systems applications, Proceedings, Seventh International Workshop, Sep. 9–10, 1996, pp. 116–121.

Bertino et al. Trigger Inheritance and Overriding in an Active Object Database System. IEEE Transactions on Knowledge and Data Engineering, v12, n4. Jul./Aug., 2000.

ANSII Standard for the Programming Language C++, First Edition ISO/IEC 14882: 1998. Date Sep. 1998.

The Annotated C++ Reference Manual ANSII Base Document, M.A. Ellis and B. Stroustrup. Date Jul. 1990.

IBM Dictionary of Computing, pp. 140, 241, 299, 728.

Microsoft Corporation, Microsoft Solutions Framework Overview A Quick Tour of the MSF Models, URL: http://channels.microsoft.com/enterprise/support/support/consult, Viewed Oct. 9, 1999.

* cited by examiner

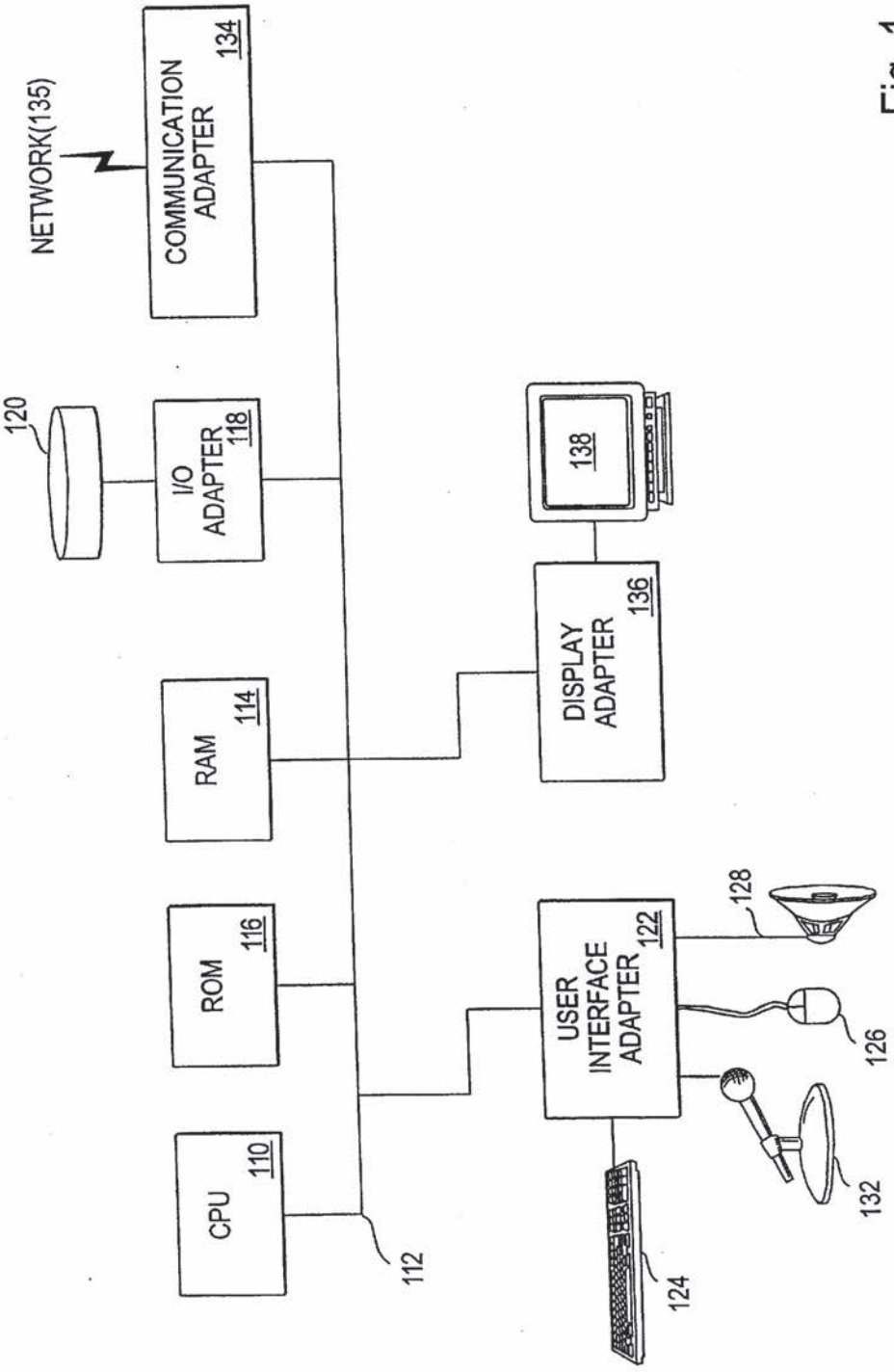

NETWORK(135)
COMMUNICATION ADAPTER 134
120
I/O ADAPTER 118
138
DISPLAY ADAPTER 136
RAM 114
ROM 116
USER INTERFACE ADAPTER 122
128
126
132
CPU 110
112
124

Fig. 1

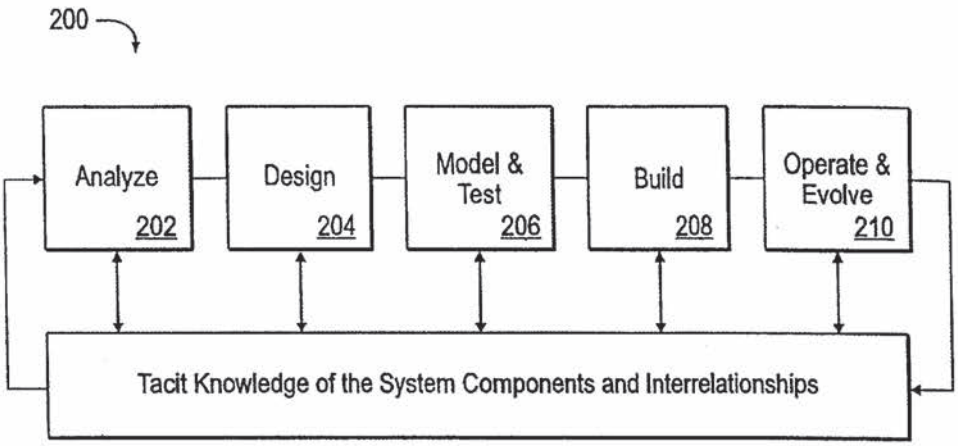

200
Analyze
202
Design
204
Model & Test
206
Build
208
Operate & Evolve
210
Tacit Knowledge of the System Components and Interrelationships

Fig. 2

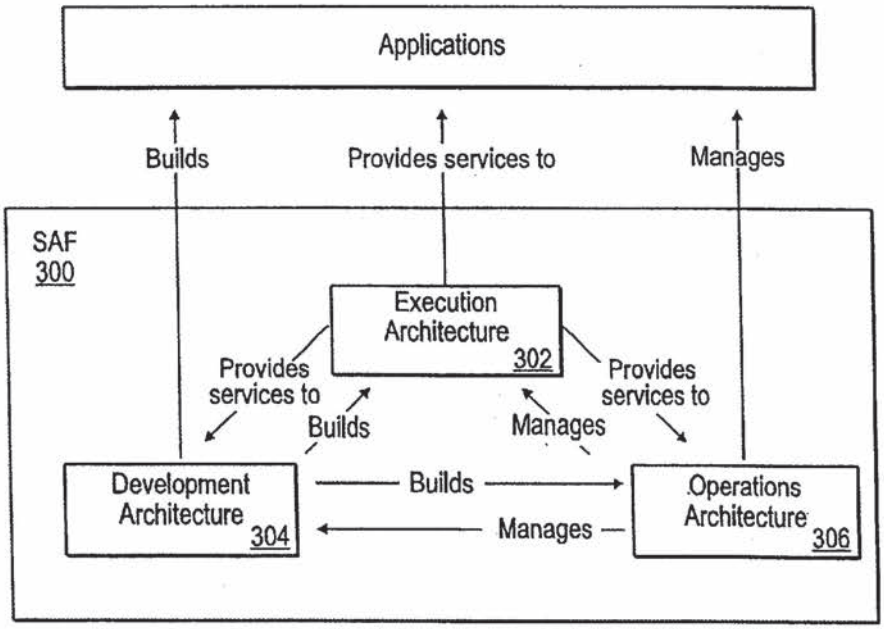

Applications
Builds
Provides services to
Manages
SAF
300
Execution Architecture
302
Provides services to
Builds
Manages
Provides services to
Development Architecture
304
Builds
Manages
Operations Architecture
306

Fig. 3

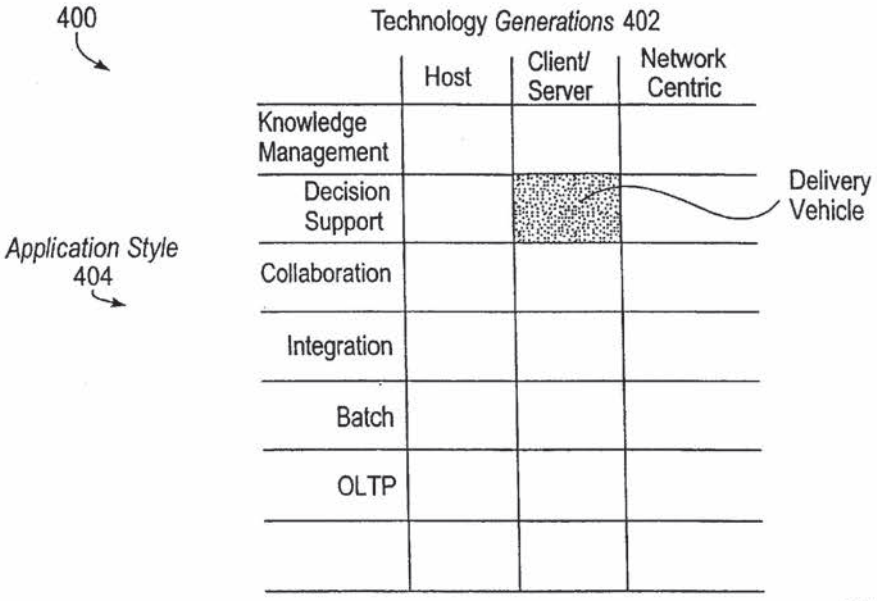

400
Technology Generations 402
Host
Client/ Server
Network Centric
Knowledge Management
Decision Support
Delivery Vehicle
Application Style 404
Collaboration
Integration
Batch
OLTP

Fig. 4

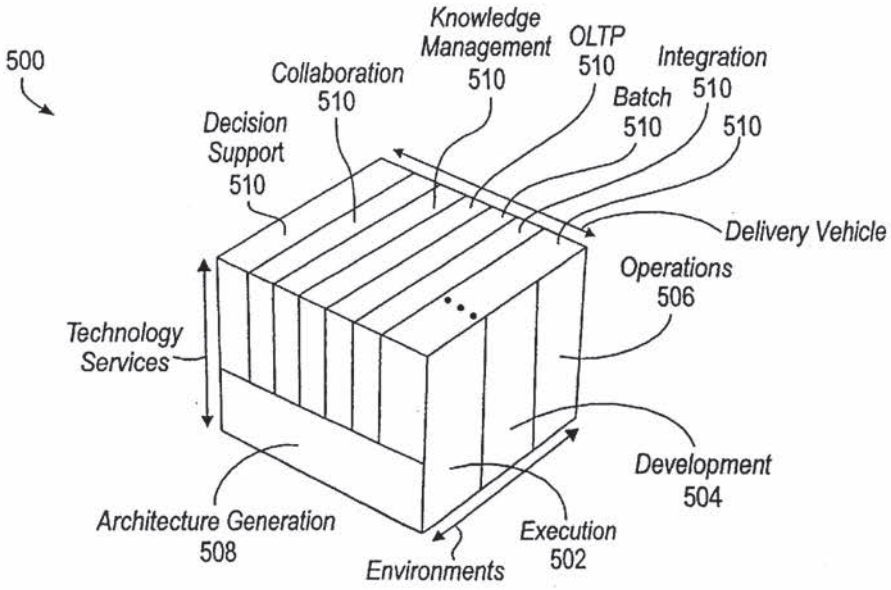

500
Knowledge Management 510
OLTP 510
Integration 510
Collaboration 510
Batch 510
510
Decision Support 510
Delivery Vehicle
Operations 506
Technology Services
Development 504
Architecture Generation 508
Execution 502
Environments

Fig. 5

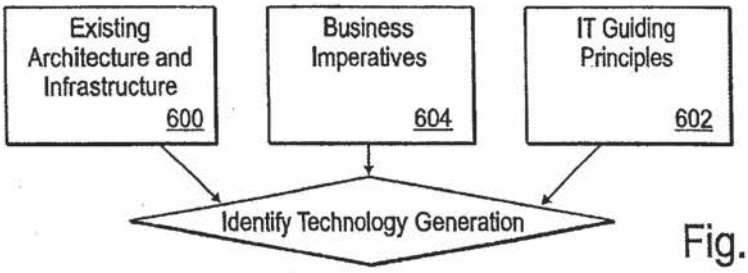

Existing Architecture and Infrastructure
600
Business Imperatives
604
IT Guiding Principles
602
Identify Technology Generation

Fig. 6

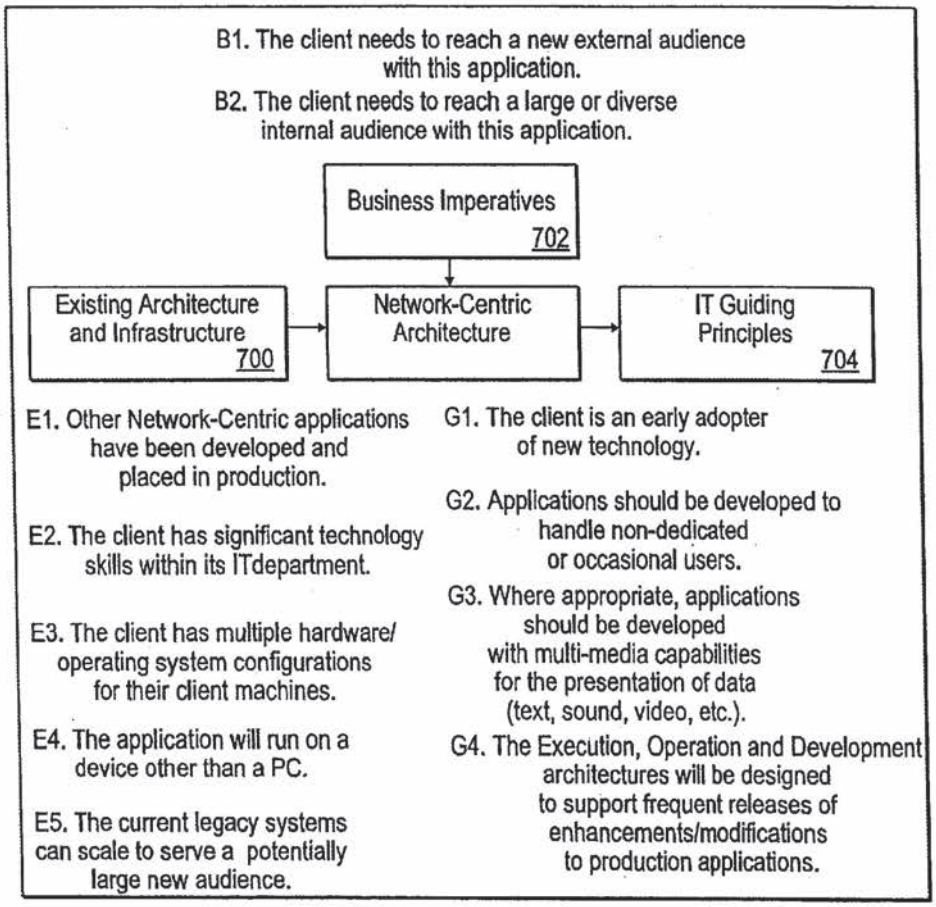

B1. The client needs to reach a new external audience with this application.
B2. The client needs to reach a large or diverse internal audience with this application.
Business Imperatives
702
Existing Architecture and Infrastructure
700
Network-Centric Architecture
IT Guiding Principles
704
E1. Other Network-Centric applications have been developed and placed in production.
E2. The client has significant technology skills within its ITdepartment.
E3. The client has multiple hardware/ operating system configurations for their client machines.
E4. The application will run on a device other than a PC.
E5. The current legacy systems can scale to serve a potentially large new audience.
G1. The client is an early adopter of new technology.
G2. Applications should be developed to handle non-dedicated or occasional users.
G3. Where appropriate, applications should be developed with multi-media capabilities for the presentation of data (text, sound, video, etc.).
G4. The Execution, Operation and Development architectures will be designed to support frequent releases of enhancements/modifications to production applications.

Fig. 7

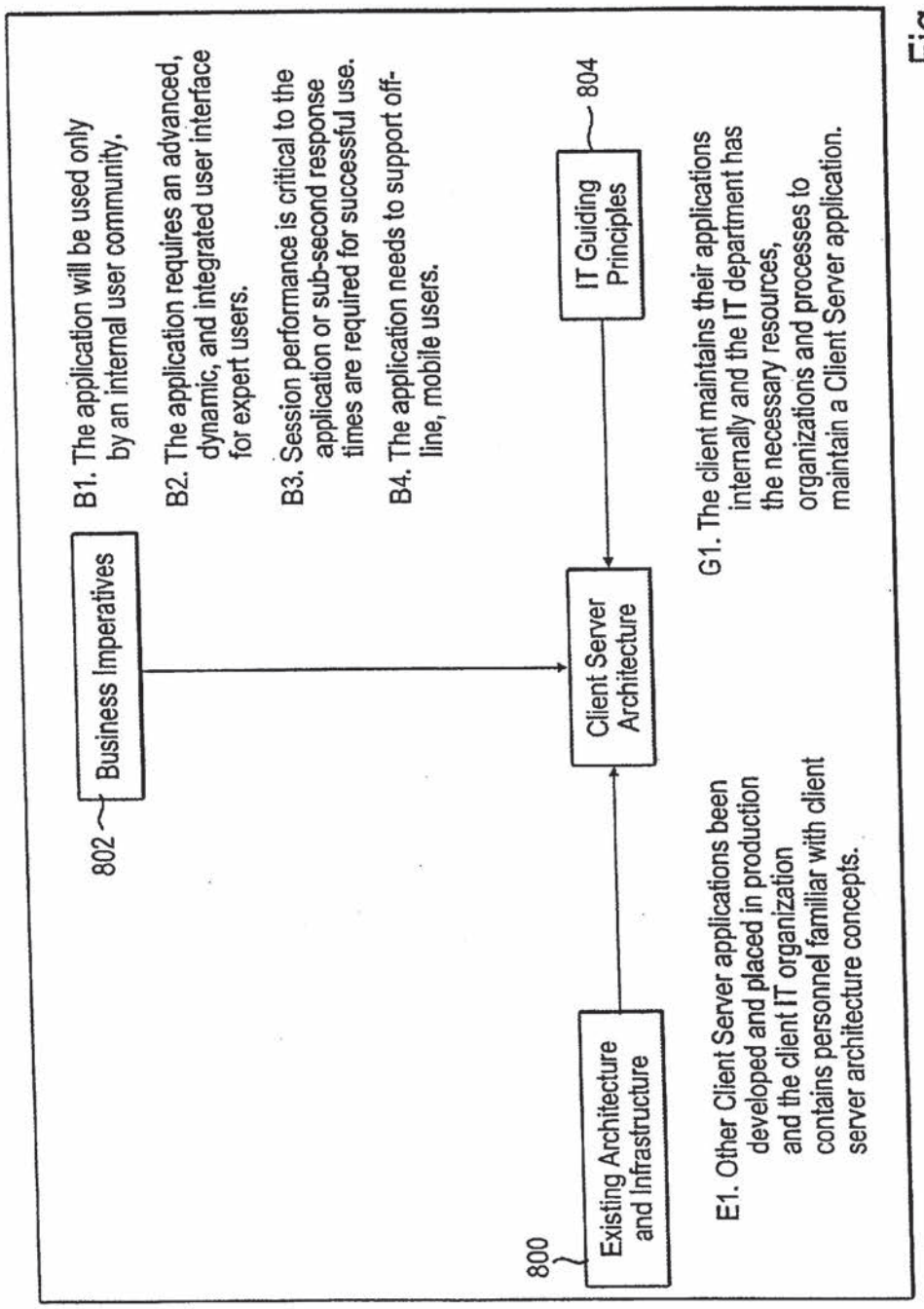

802
Business Imperatives
B1. The application will be used only by an internal user community.
B2. The application requires an advanced, dynamic, and integrated user interface for expert users.
B3. Session performance is critical to the application or sub-second response times are required for successful use.
B4. The application needs to support off-line, mobile users.
804
IT Guiding Principles
Client Server Architecture
G1. The client maintains their applications internally and the IT department has the necessary resources, organizations and processes to maintain a Client Server application.
800
Existing Architecture and Infrastructure
E1. Other Client Server applications been developed and placed in production and the client IT organization contains personnel familiar with client server architecture concepts.

Fig. 8

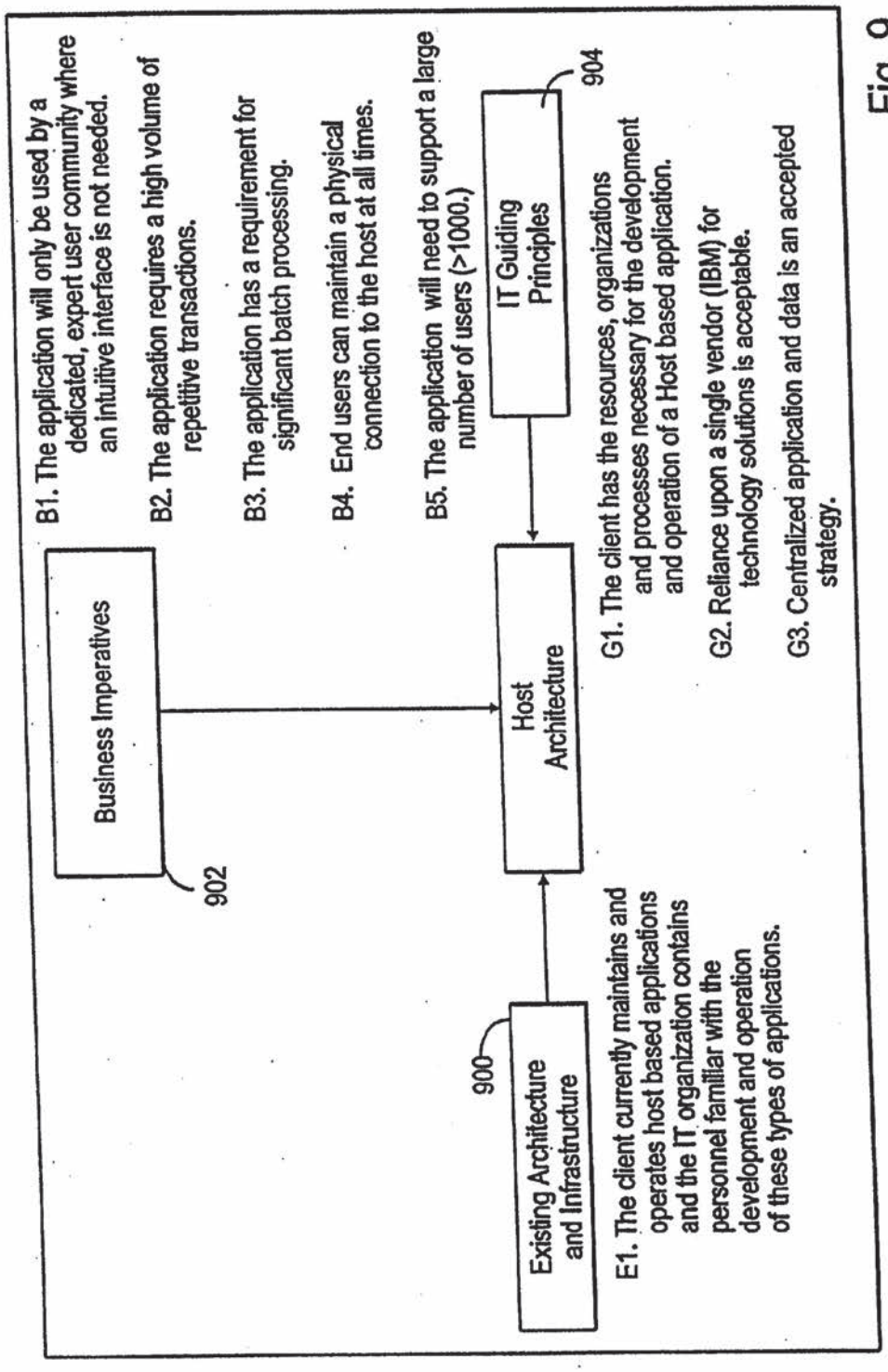

Business Imperatives
902
B1. The application will only be used by a dedicated, expert user community where an intuitive interface is not needed.
B2. The application requires a high volume of repetitive transactions.
B3. The application has a requirement for significant batch processing.
B4. End users can maintain a physical connection to the host at all times.
B5. The application will need to support a large number of users (>1000.)
IT Guiding Principles
904
Host Architecture
G1. The client has the resources, organizations and processes necessary for the development and operation of a Host based application.
G2. Reliance upon a single vendor (IBM) for technology solutions is acceptable.
G3. Centralized application and data is an accepted strategy.
900
Existing Architecture and Infrastructure
E1. The client currently maintains and operates host based applications and the IT organization contains personnel familiar with the development and operation of these types of applications.

Fig. 9

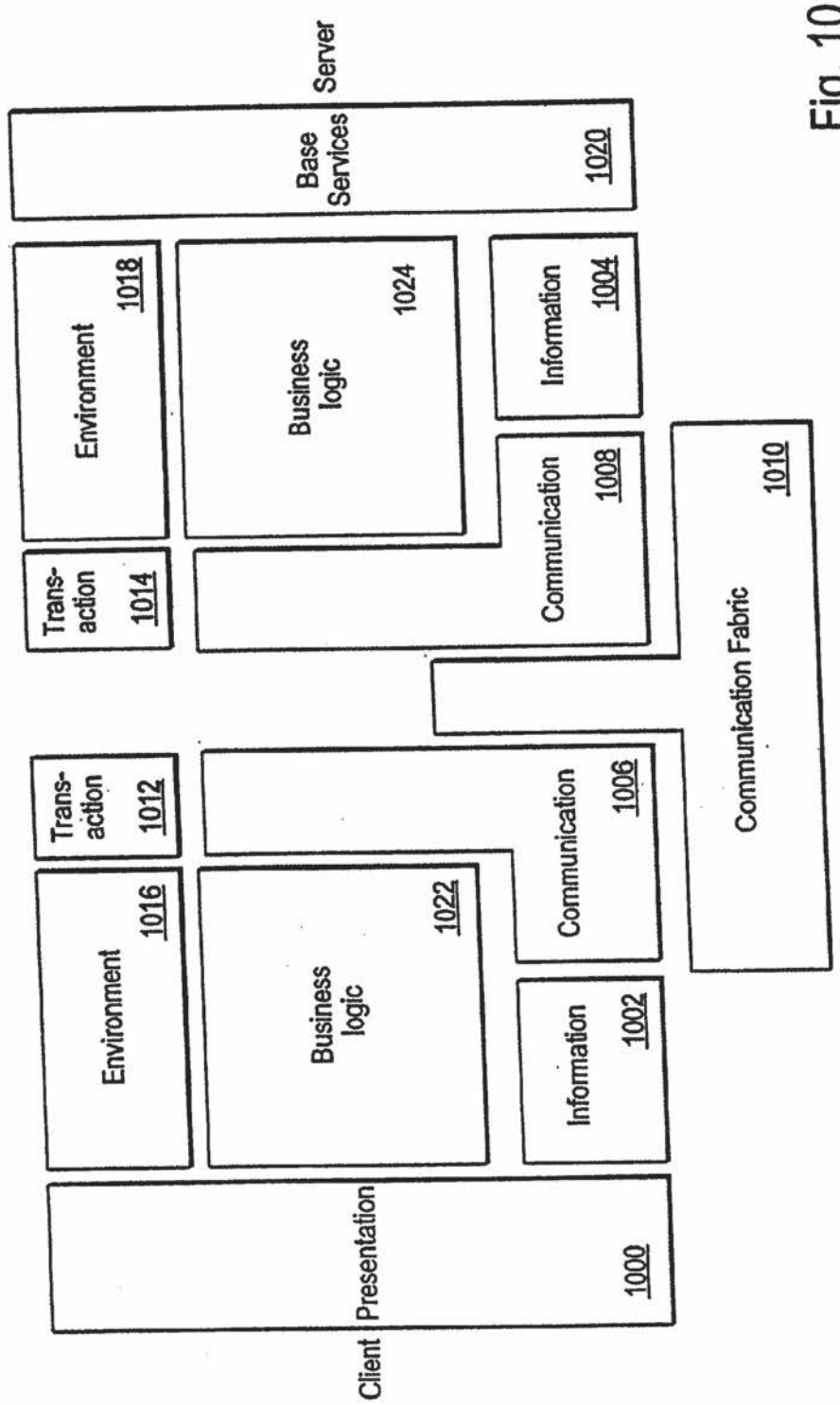

Client
Server
Presentation 1000
Information 1002
Business logic 1022
Environment 1016
Trans-action 1012
Communication 1006
Communication Fabric 1010
Communication 1008
Information 1004
Business logic 1024
Environment 1018
Trans-action 1014
Base Services 1020

Fig. 10

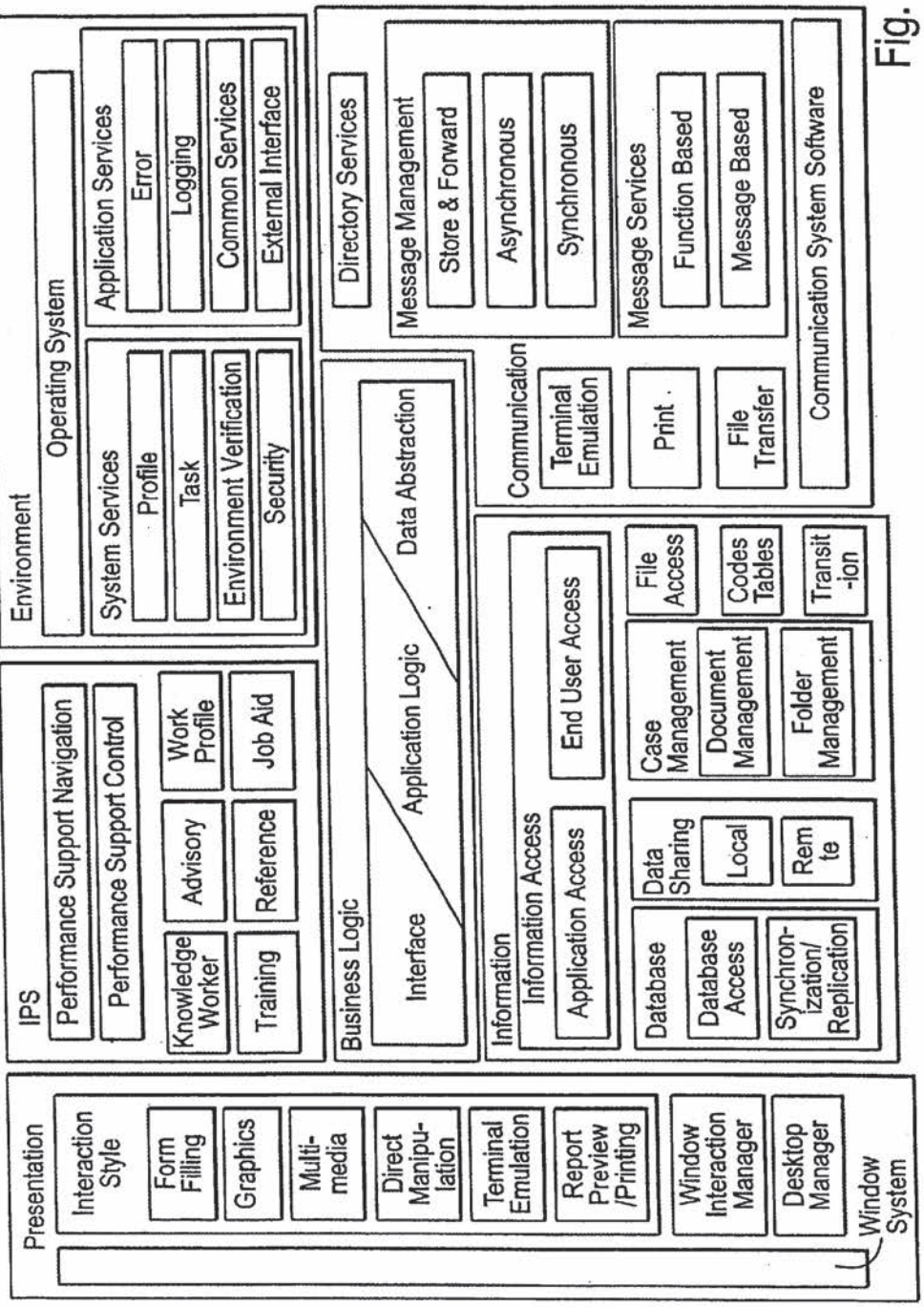

Presentation
Interaction Style
Form Filling
Graphics
Multi-media
Direct Manipu-lation
Terminal Emulation
Report Preview /Printing
Window Interaction Manager
Desktop Manager
Window System
IPS
Performance Support Navigation
Performance Support Control
Knowledge Worker
Advisory
Work Profile
Training
Reference
Job Aid
Environment
Operating System
System Services
Profile
Task
Environment Verification
Security
Application Services
Error
Logging
Common Services
External Interface
Business Logic
Interface
Application Logic
Data Abstraction
Information
Information Access
Application Access
End User Access
Database
Database Access
Synchron-ization/ Replication
Data Sharing
Local
Rem te
Case Management
Document Management
Folder Management
File Access
Codes Tables
Transit -ion
Communication
Terminal Emulation
Print
File Transfer
Communication System Software
Directory Services
Message Management
Store & Forward
Asynchronous
Synchronous
Message Services
Function Based
Message Based

Fig. 11

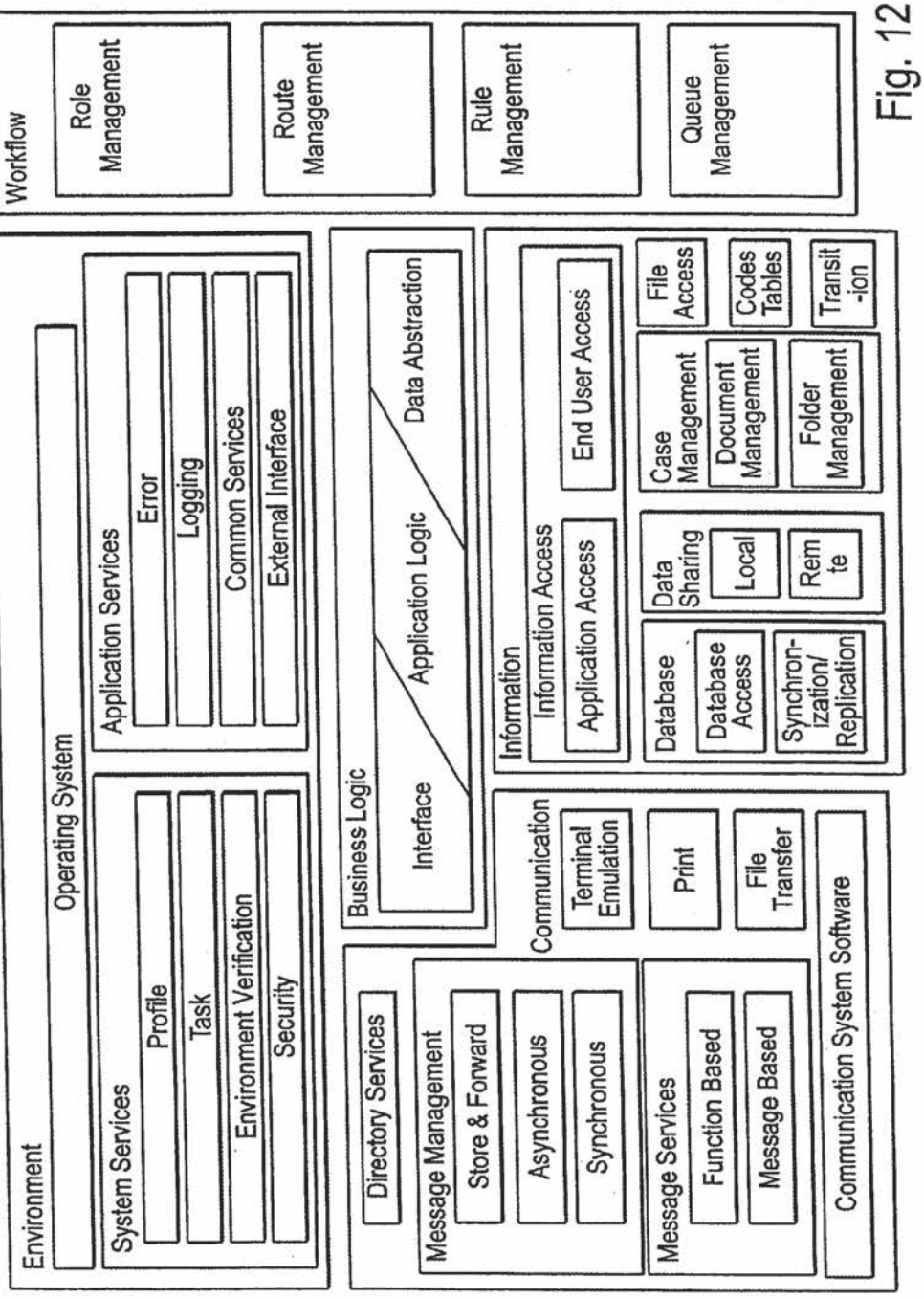

Workflow
Role Management
Route Management
Rule Management
Queue Management
Environment
Operating System
System Services
Profile
Task
Environment Verification
Security
Application Services
Error
Logging
Common Services
External Interface
Business Logic
Interface
Application Logic
Data Abstraction
Information
Information Access
Application Access
End User Access
Database
Database Access
Synchron-ization/ Replication
Data Sharing
Local
Rem te
Case Management
Document Management
Folder Management
File Access
Codes Tables
Transit -ion
Directory Services
Message Management
Store & Forward
Asynchronous
Synchronous
Communication
Terminal Emulation
Print
File Transfer
Message Services
Function Based
Message Based
Communication System Software

Fig. 12

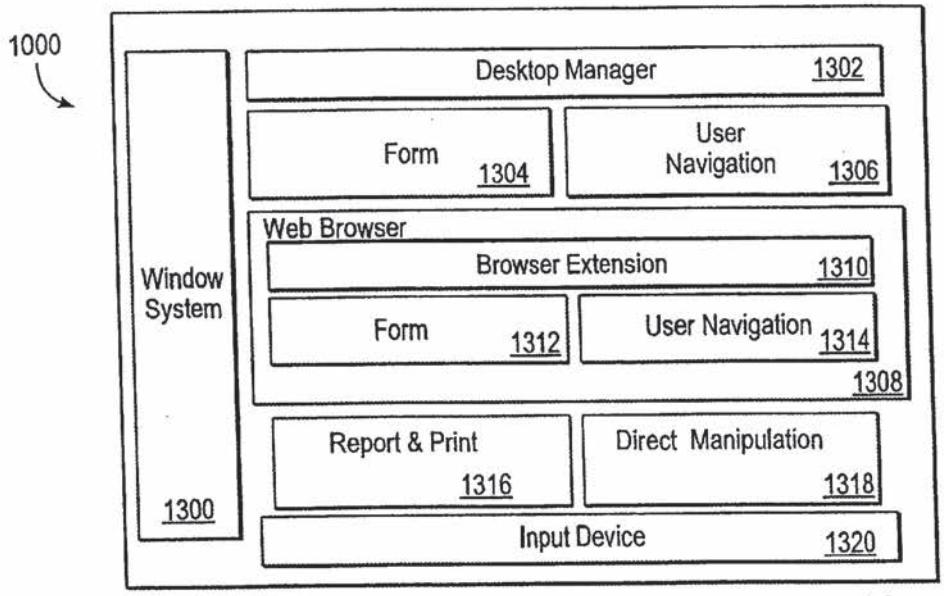

1000
Desktop Manager 1302
Form 1304
User Navigation 1306
Web Browser
Browser Extension 1310
Form 1312
User Navigation 1314
1308
Window System
1300
Report & Print 1316
Direct Manipulation 1318
Input Device 1320

Fig. 13

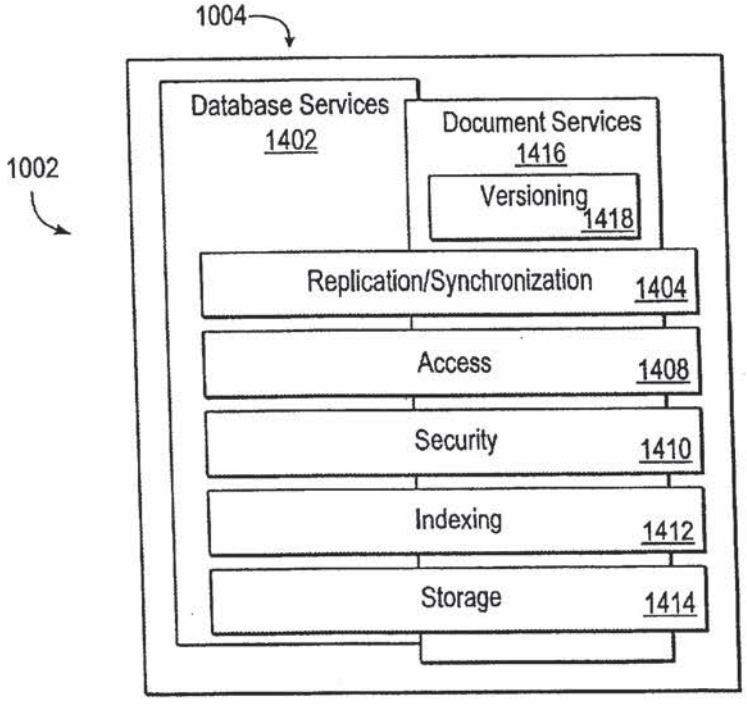

1004
1002
Database Services 1402
Document Services 1416
Versioning 1418
Replication/Synchronization 1404
Access 1408
Security 1410
Indexing 1412
Storage 1414

Fig. 14

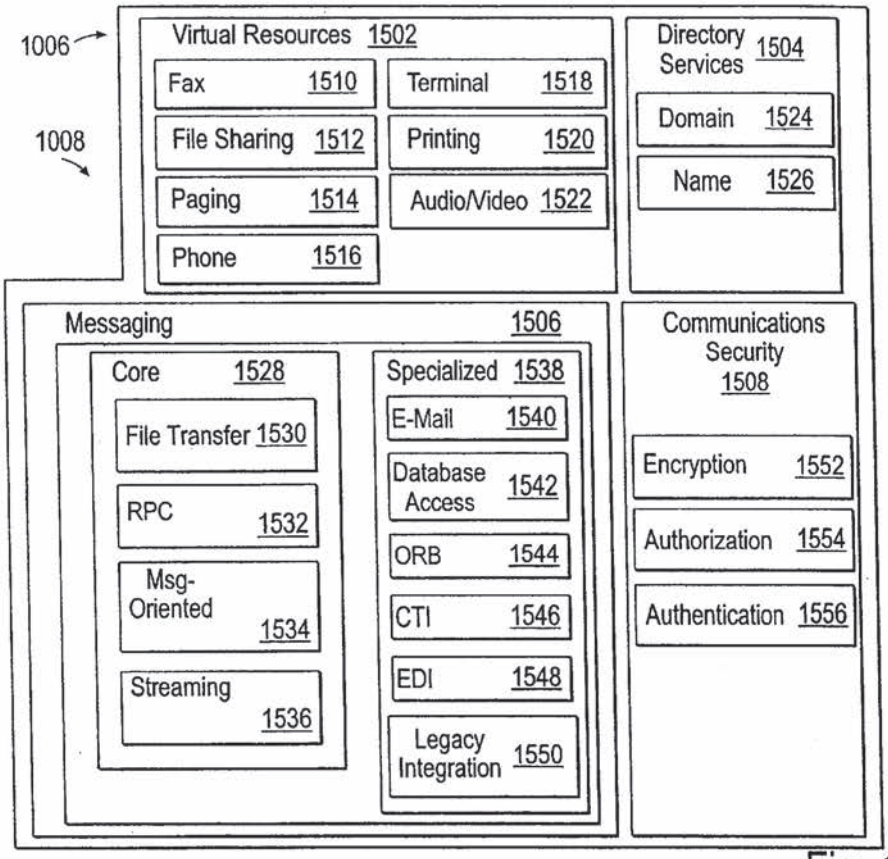

1006
1008
Virtual Resources 1502
Fax 1510
Terminal 1518
File Sharing 1512
Printing 1520
Paging 1514
Audio/Video 1522
Phone 1516
Directory Services 1504
Domain 1524
Name 1526
Messaging 1506
Core 1528
File Transfer 1530
RPC 1532
Msg-Oriented 1534
Streaming 1536
Specialized 1538
E-Mail 1540
Database Access 1542
ORB 1544
CTI 1546
EDI 1548
Legacy Integration 1550
Communications Security 1508
Encryption 1552
Authorization 1554
Authentication 1556

Fig. 15

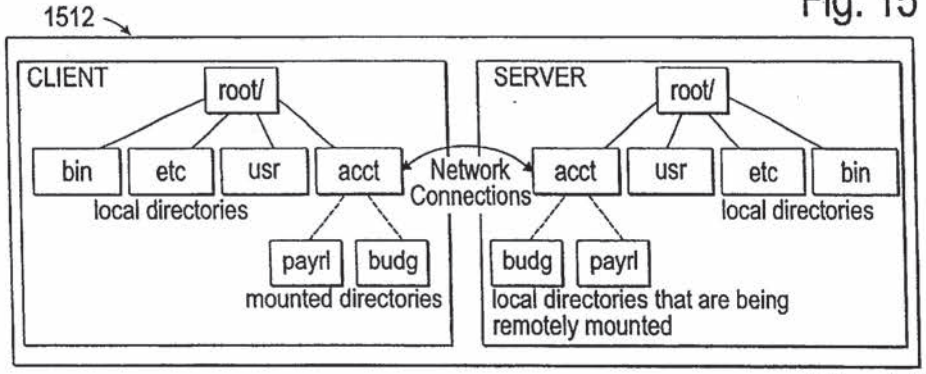

1512
CLIENT
root/
bin
etc
usr
acct
local directories
payrl
budg
mounted directories
Network Connections
SERVER
root/
acct
usr
etc
bin
local directories
budg
payrl
local directories that are being remotely mounted

Fig. 16



Fig. 17

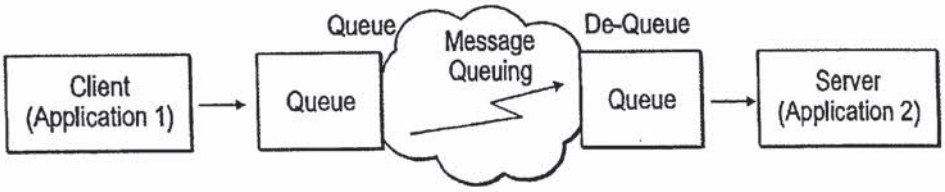


Fig. 18



Fig. 19

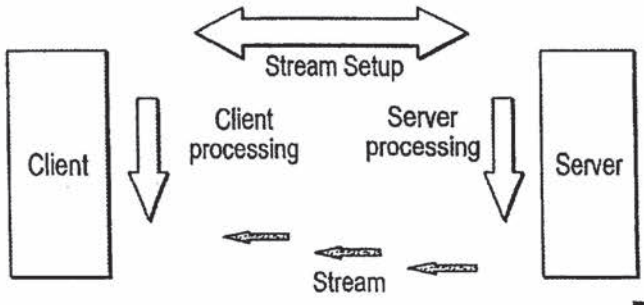


Fig. 20

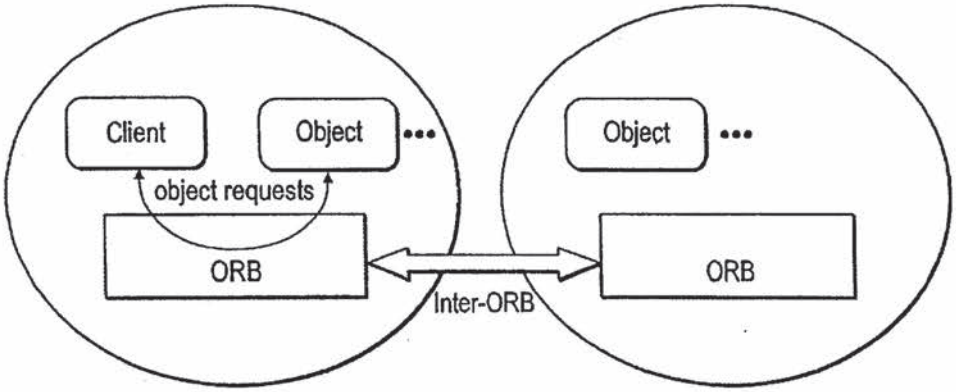


Fig. 21

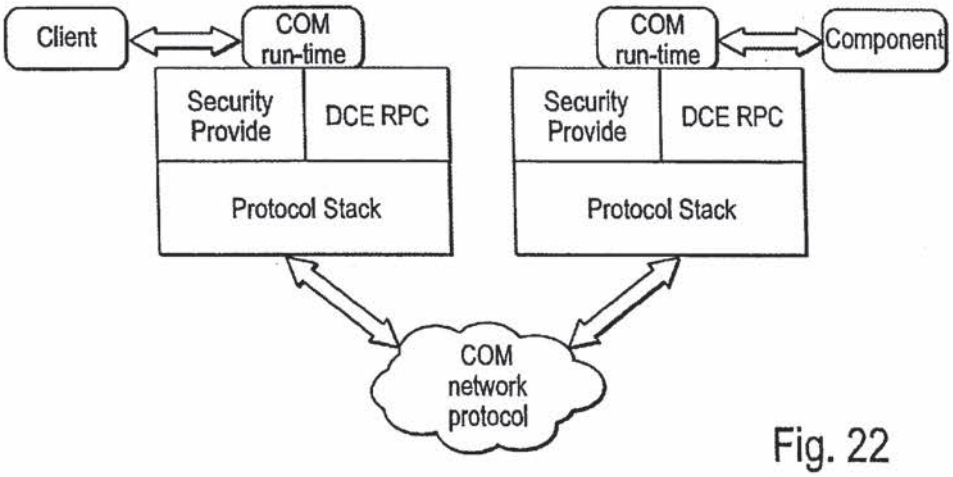


Fig. 22

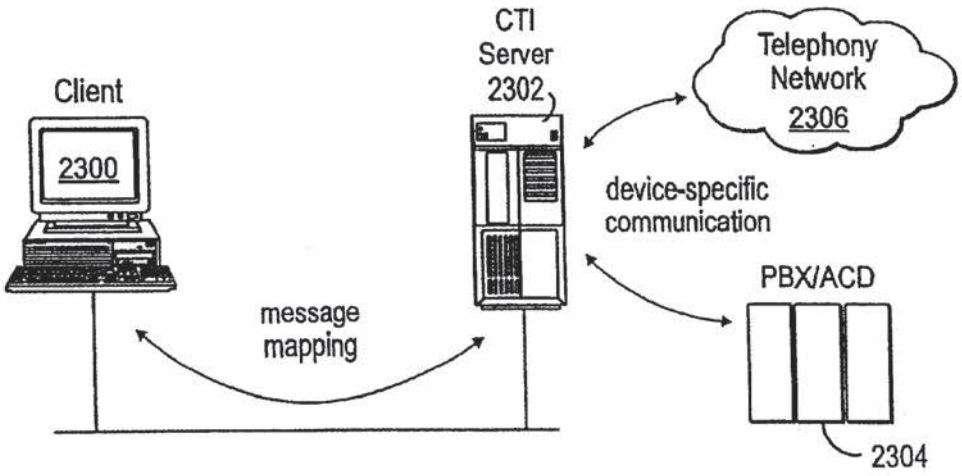

CTI
Server
2302
Telephony
Network
2306
Client
2300
device-specific
communication
PBX/ACD
message
mapping
2304

Fig. 23

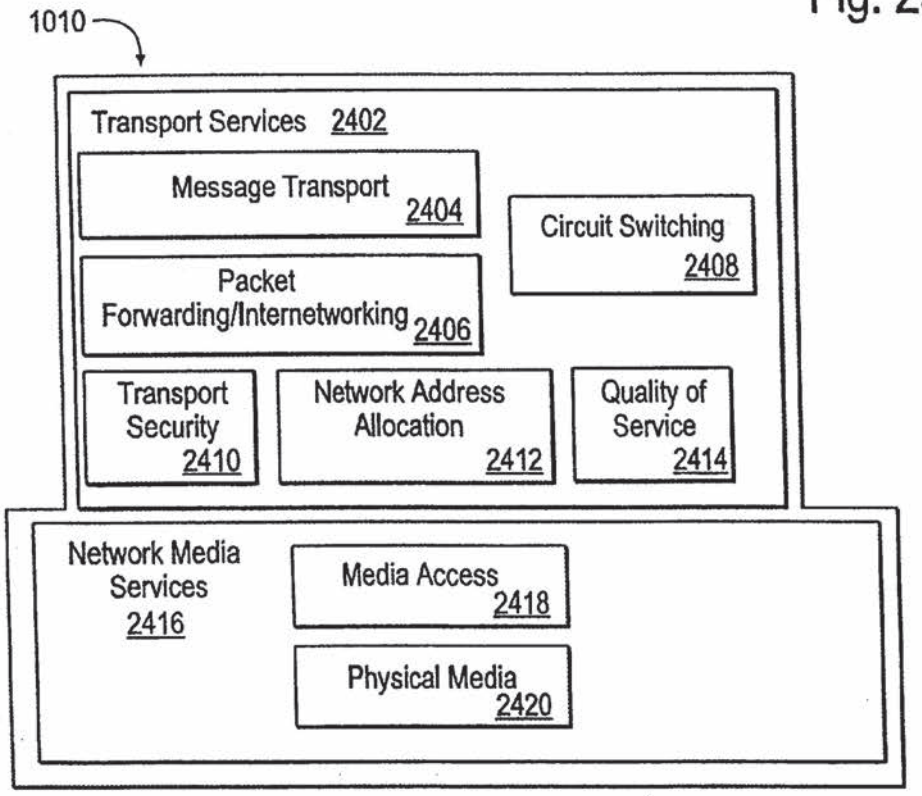

1010
Transport Services 2402
Message Transport 2404
Circuit Switching 2408
Packet
Forwarding/Internetworking 2406
Transport
Security
2410
Network Address
Allocation
2412
Quality of
Service
2414
Network Media
Services
2416
Media Access
2418
Physical Media
2420

Fig. 24

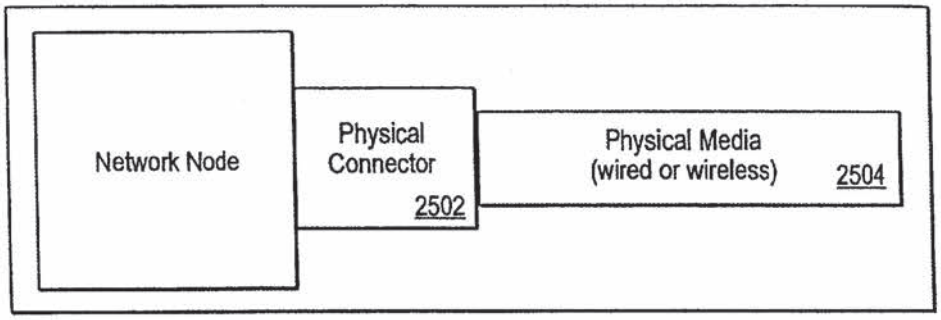

Network Node
Physical Connector
2502
Physical Media (wired or wireless)
2504

Fig. 25

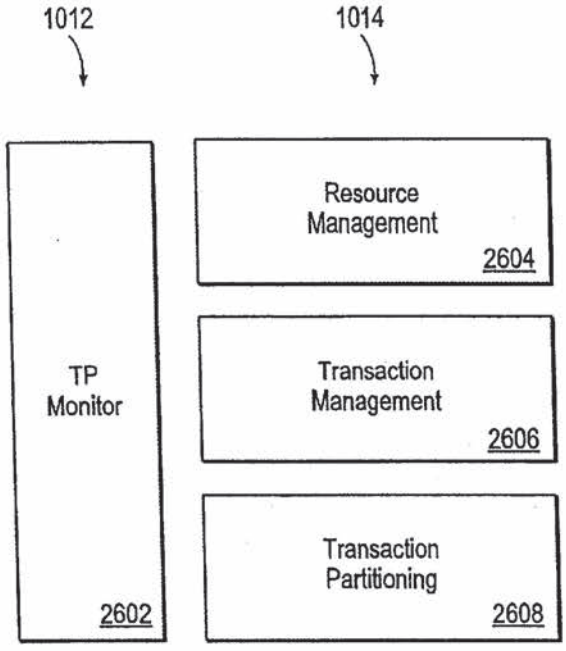

1012
1014
TP Monitor
2602
Resource Management
2604
Transaction Management
2606
Transaction Partitioning
2608

Fig. 26

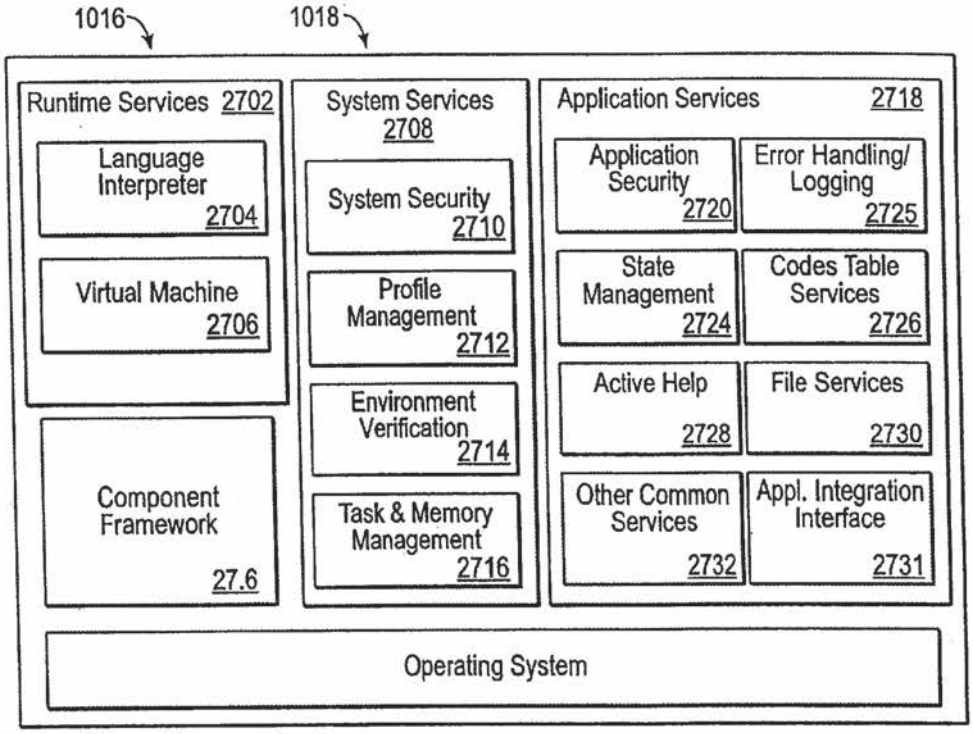

1016
1018
Runtime Services 2702
Language Interpreter 2704
Virtual Machine 2706
Component Framework 27.6
System Services 2708
System Security 2710
Profile Management 2712
Environment Verification 2714
Task & Memory Management 2716
Application Services 2718
Application Security 2720
Error Handling/ Logging 2725
State Management 2724
Codes Table Services 2726
Active Help 2728
File Services 2730
Other Common Services 2732
Appl. Integration Interface 2731
Operating System

Fig. 27

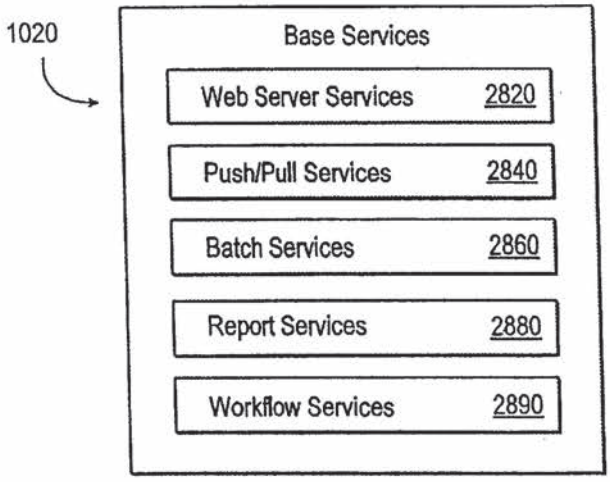

1020
Base Services
Web Server Services 2820
Push/Pull Services 2840
Batch Services 2860
Report Services 2880
Workflow Services 2890

Fig. 28

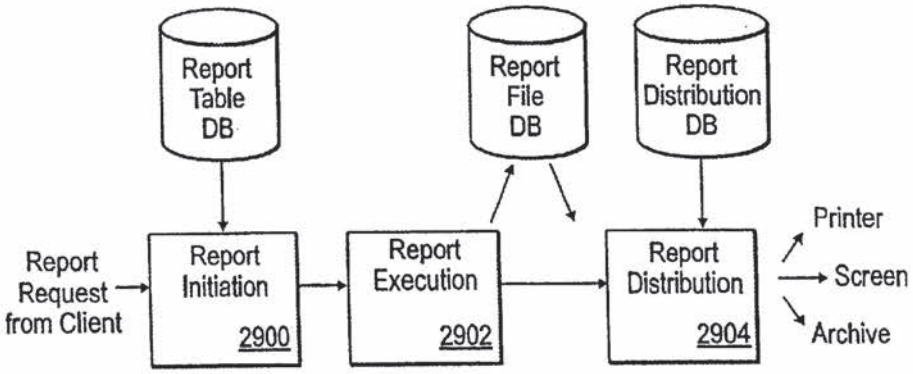

Report Table DB
Report File DB
Report Distribution DB
Report Request from Client
Report Initiation 2900
Report Execution 2902
Report Distribution 2904
Printer
Screen
Archive

Fig. 29

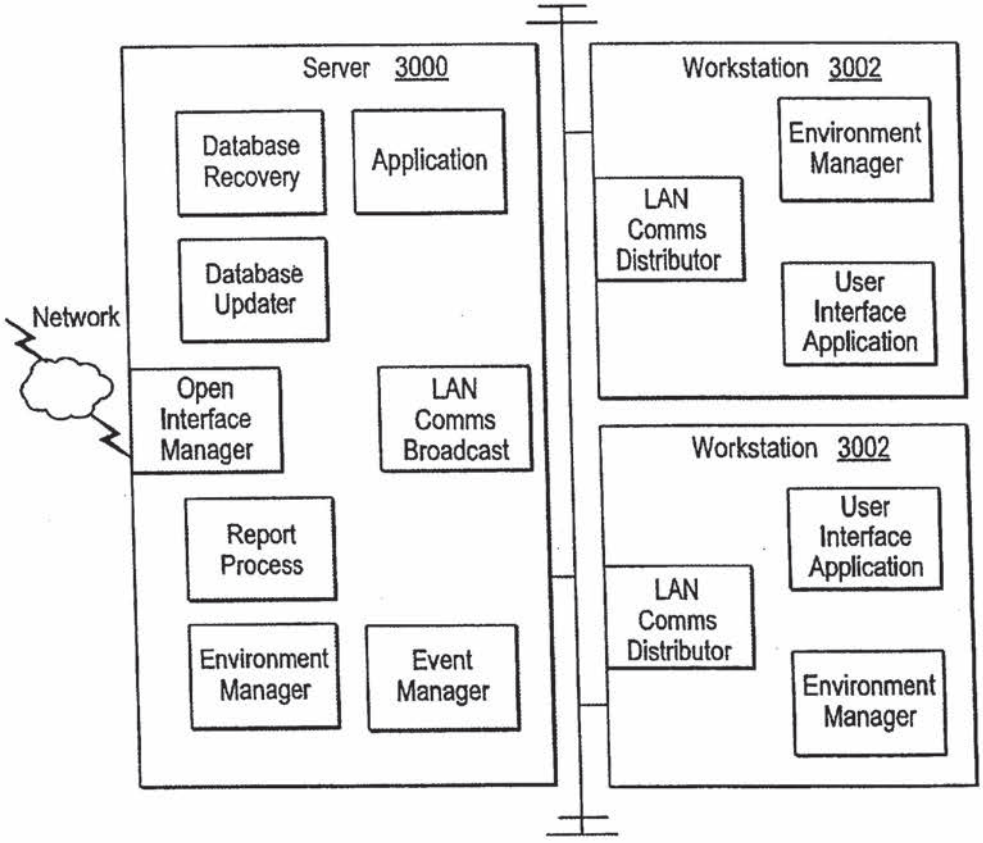

Server 3000
Database Recovery
Application
Database Updater
Network
Open Interface Manager
LAN Comms Broadcast
Report Process
Environment Manager
Event Manager
Workstation 3002
Environment Manager
LAN Comms Distributor
User Interface Application
Workstation 3002
User Interface Application
LAN Comms Distributor
Environment Manager

Fig. 30

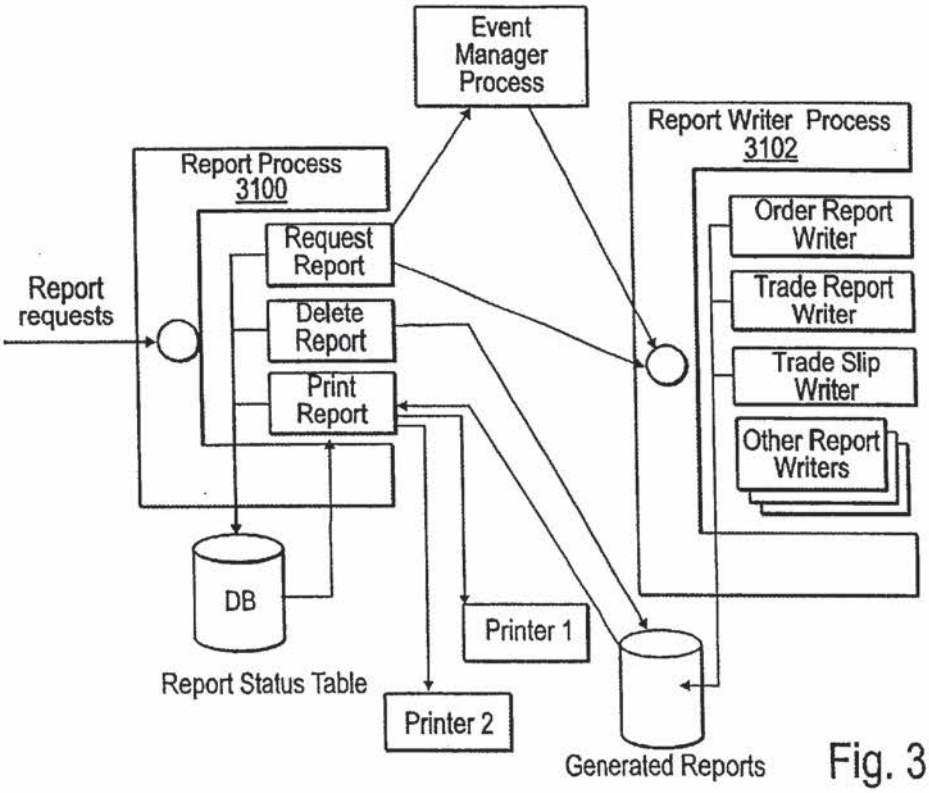

Event Manager Process
Report Writer Process 3102
Report Process 3100
Request Report
Report requests
Delete Report
Print Report
Order Report Writer
Trade Report Writer
Trade Slip Writer
Other Report Writers
DB
Report Status Table
Printer 1
Printer 2
Generated Reports

Fig. 31

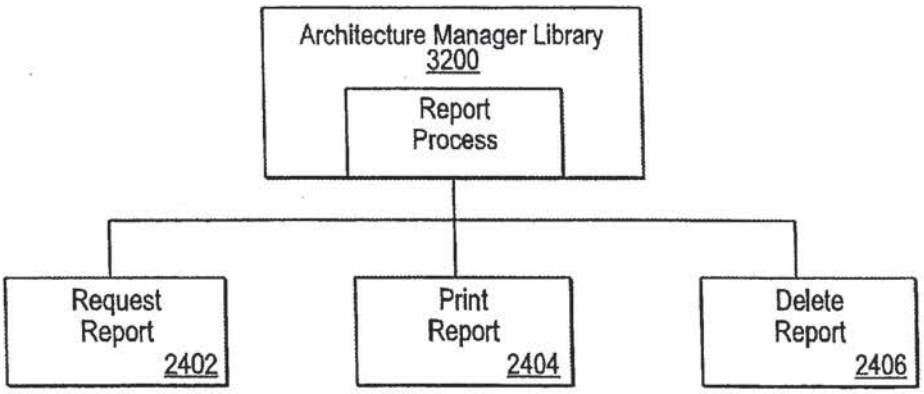

Architecture Manager Library 3200
Report Process
Request Report 2402
Print Report 2404
Delete Report 2406

Fig. 32

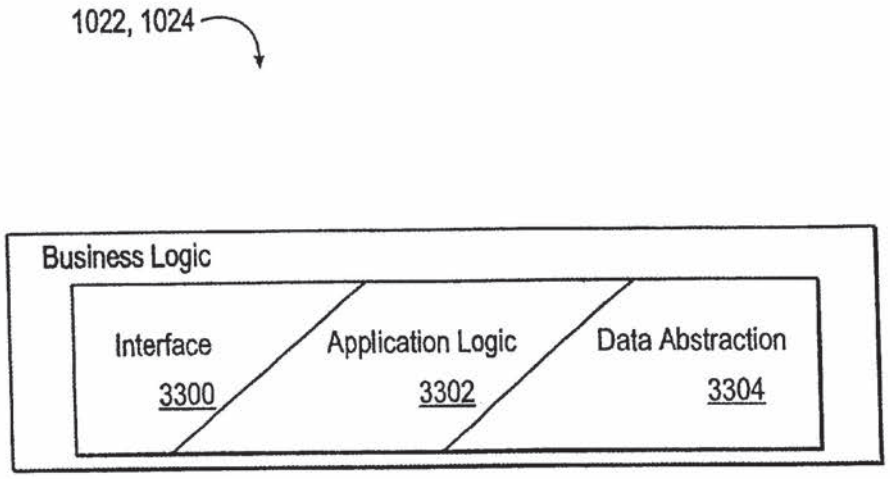

1022, 1024
Business Logic
Interface
3300
Application Logic
3302
Data Abstraction
3304

Fig. 33

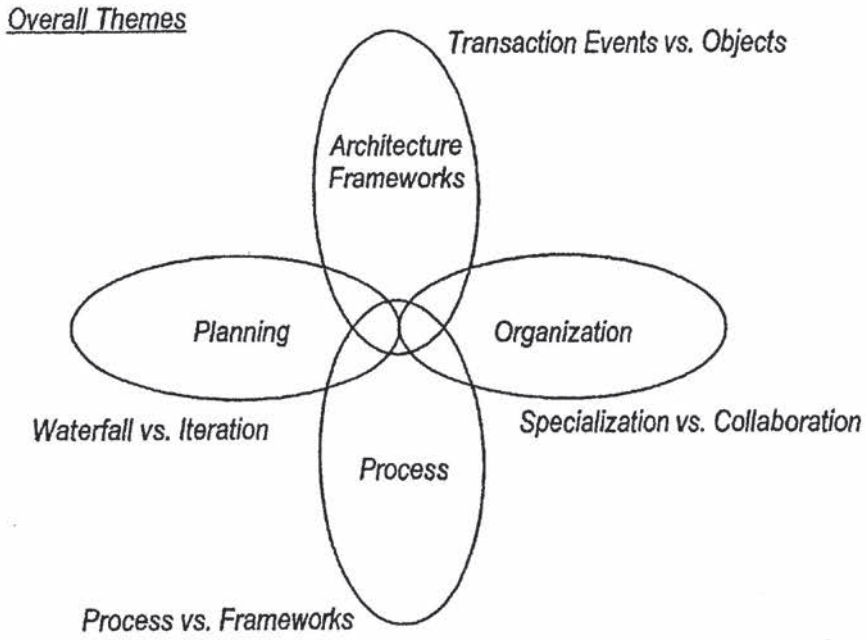

Overall Themes
Transaction Events vs. Objects
Architecture Frameworks
Planning
Organization
Waterfall vs. Iteration
Specialization vs. Collaboration
Process
Process vs. Frameworks

Fig. 34

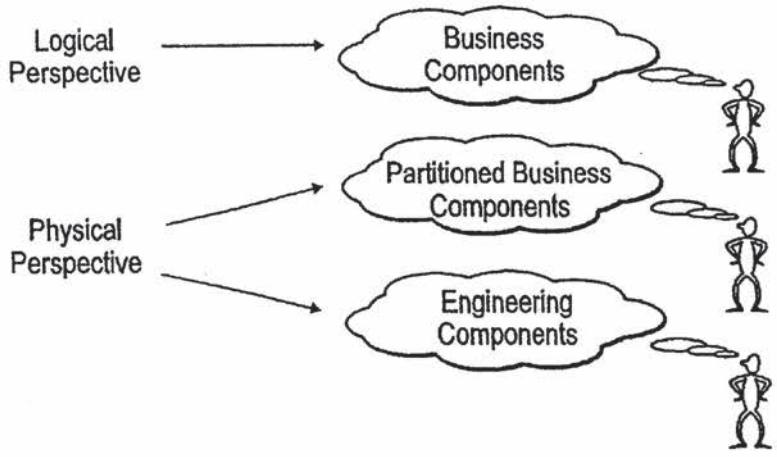

Logical Perspective
Business Components
Partitioned Business Components
Physical Perspective
Engineering Components

Fig. 35

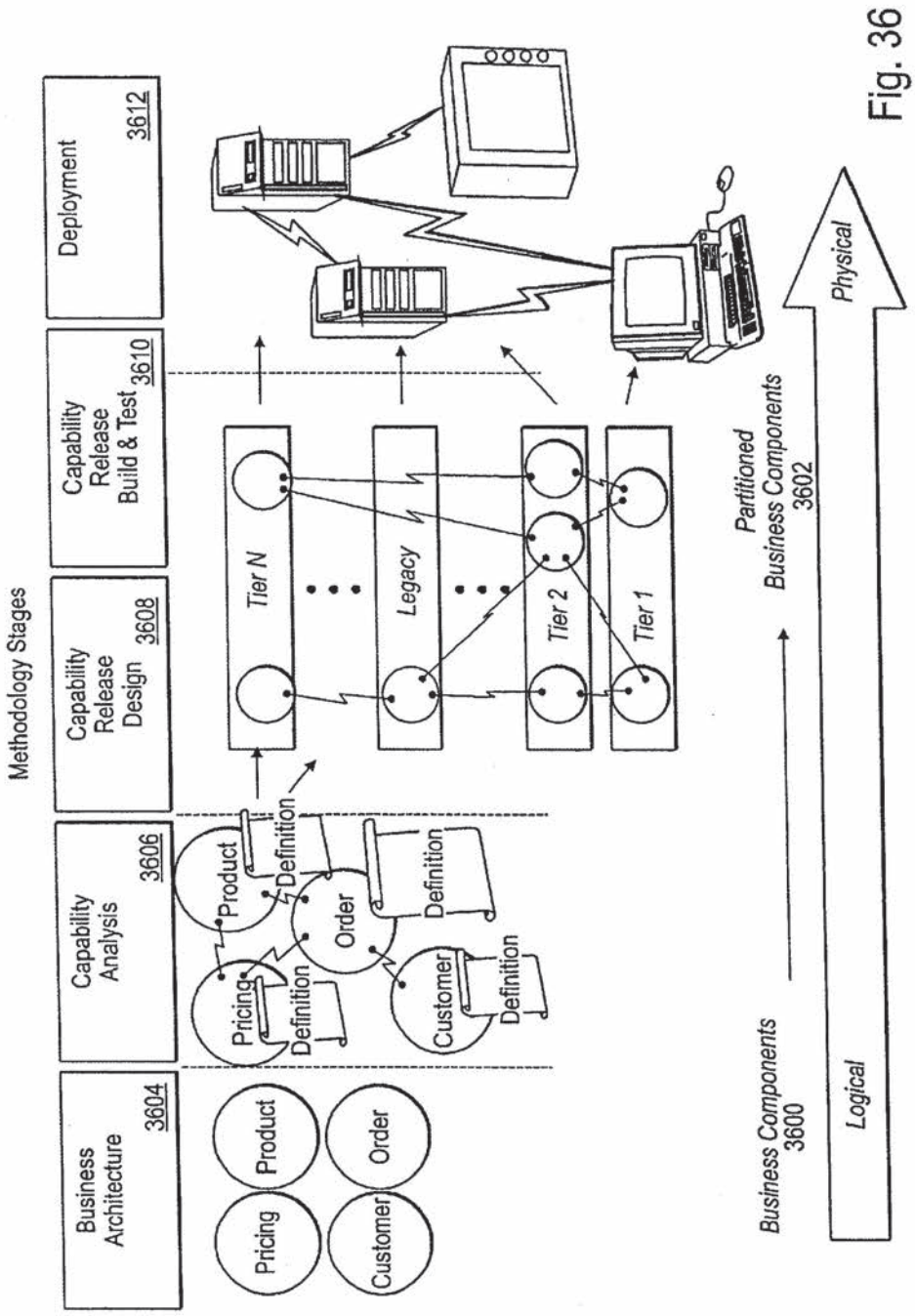

Methodology Stages
Business Architecture 3604
Capability Analysis 3606
Capability Release Design 3608
Capability Release Build & Test 3610
Deployment 3612
Pricing
Product
Order
Customer
Pricing Definition
Product Definition
Order Definition
Customer Definition
Tier N
Legacy
Tier 2
Tier 1
Partitioned Business Components 3602
Business Components 3600
Logical
Physical

Fig. 36

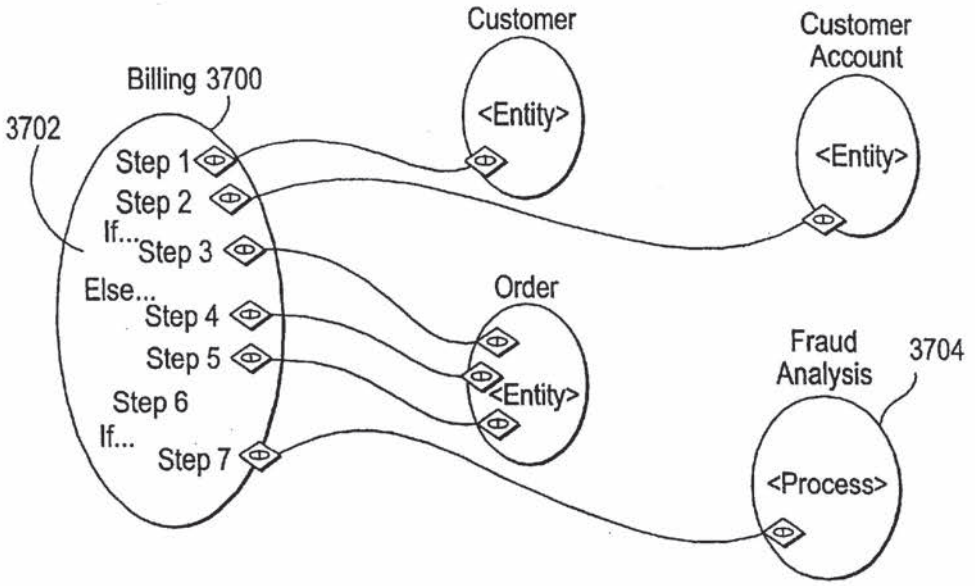

Customer
Customer Account
Billing 3700
3702
<Entity>
<Entity>
Step 1
Step 2
If...
Step 3
Else...
Step 4
Step 5
Step 6
If...
Step 7
Order
<Entity>
Fraud Analysis
3704
<Process>

Fig. 37

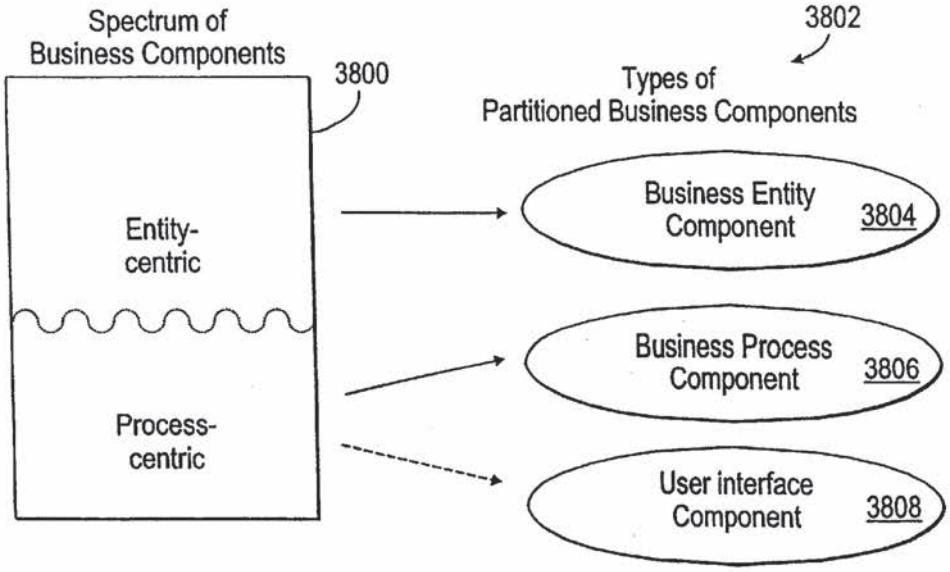

Spectrum of Business Components
3800
3802
Types of Partitioned Business Components
Entity-centric
Process-centric
Business Entity Component
3804
Business Process Component
3806
User interface Component
3808

Fig. 38

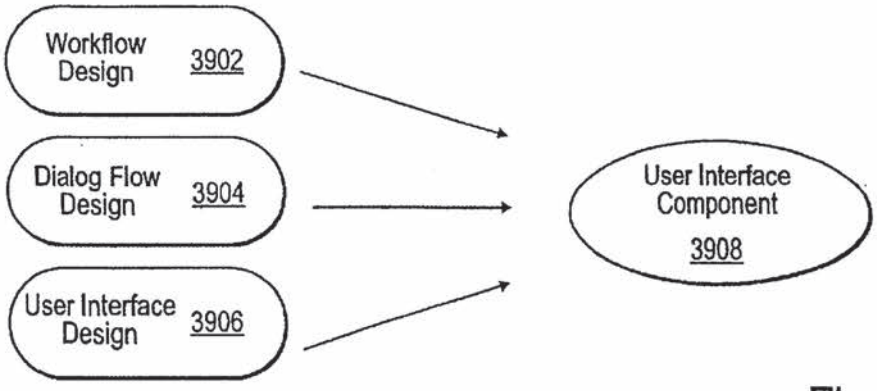

Workflow Design 3902
Dialog Flow Design 3904
User Interface Design 3906
User Interface Component 3908

Fig. 39

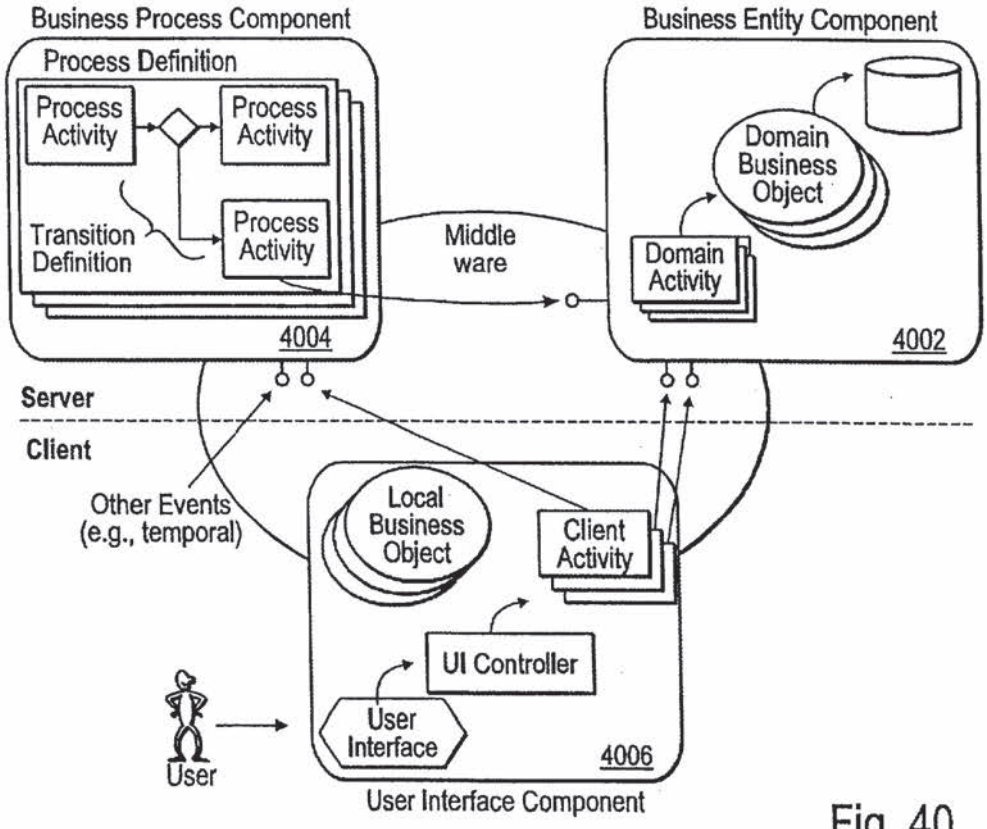
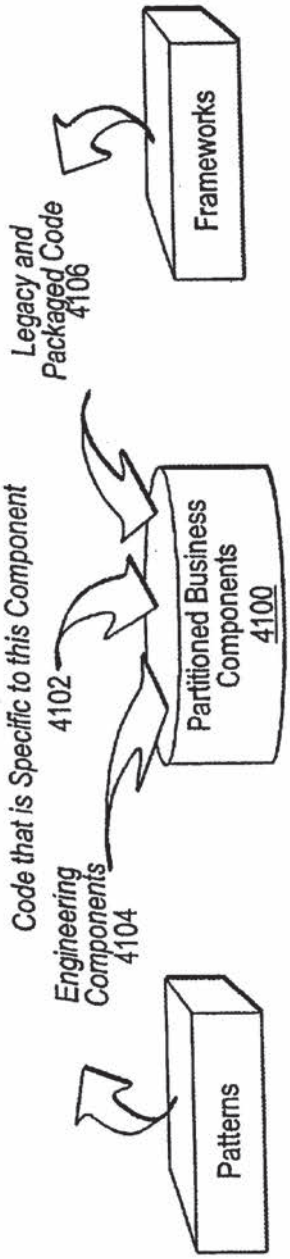

Business Process Component
Process Definition
Process Activity
Process Activity
Process Activity
Transition Definition
4004
Middle ware
Business Entity Component
Domain Business Object
Domain Activity
4002
Server
Client
Other Events (e.g., temporal)
Local Business Object
Client Activity
UI Controller
User Interface
4006
User
User Interface Component

Fig. 40

Fig. 41


Frameworks
Legacy and Packaged Code 4106
Partitioned Business Components 4100
Code that is Specific to this Component 4102
Engineering Components 4104
Patterns

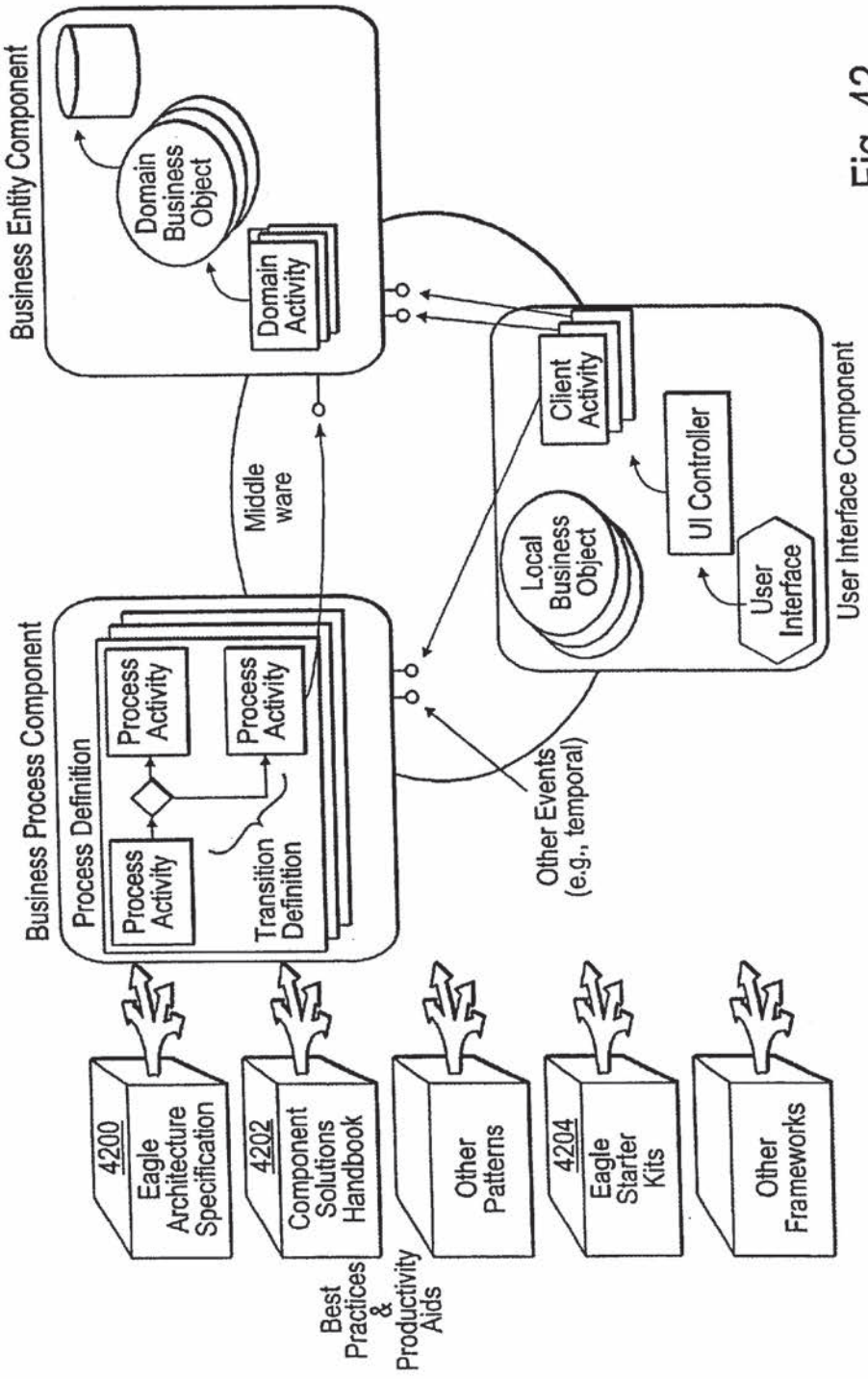

Business Entity Component
Domain Business Object
Domain Activity
Middle ware
User Interface Component
Client Activity
UI Controller
Local Business Object
User Interface
Business Process Component
Process Definition
Process Activity
Process Activity
Process Activity
Transition Definition
Other Events (e.g., temporal)
4200 Eagle Architecture Specification
4202 Component Solutions Handbook
Other Patterns
4204 Eagle Starter Kits
Other Frameworks
Best Practices & Productivity Aids

Fig. 42

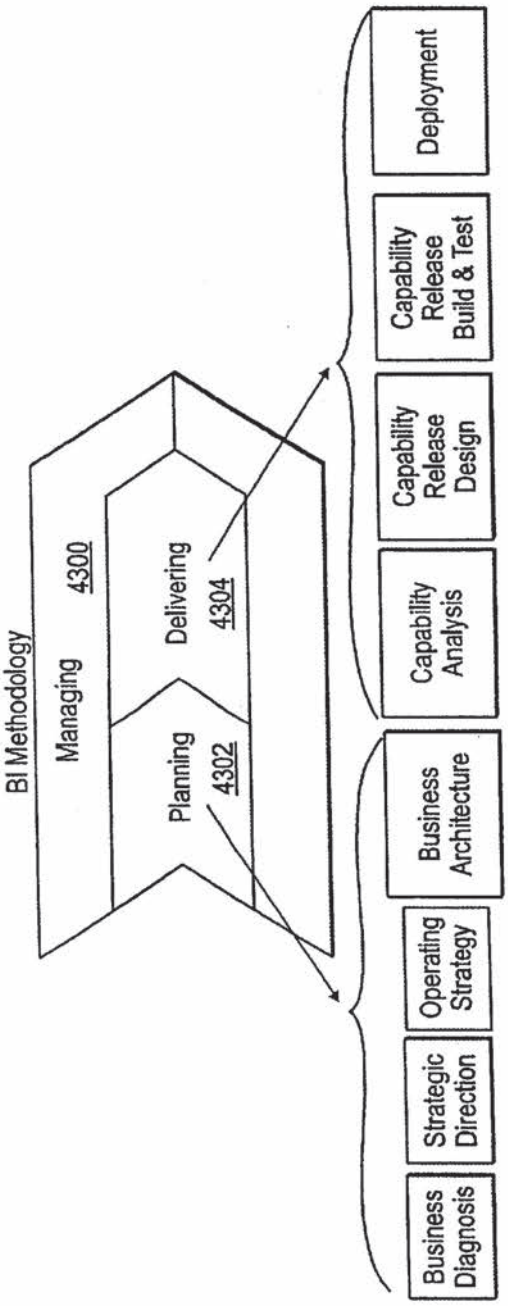

BI Methodology
Managing 4300
Planning 4302
Delivering 4304
Business Diagnosis
Strategic Direction
Operating Strategy
Business Architecture
Capability Analysis
Capability Release Design
Capability Release Build & Test
Deployment
Identify Business Components
Iteration #1
Iteration #2

Fig. 43

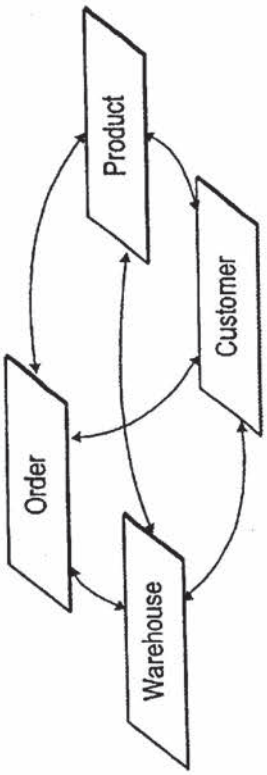

Product
Order
Customer
Warehouse

Fig. 44

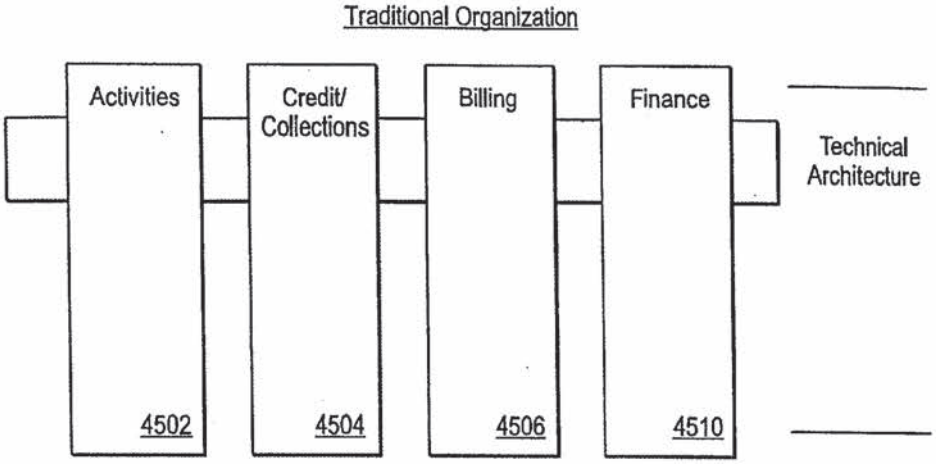

Traditional Organization
Activities
Credit/ Collections
Billing
Finance
Technical Architecture
4502
4504
4506
4510

Fig. 45

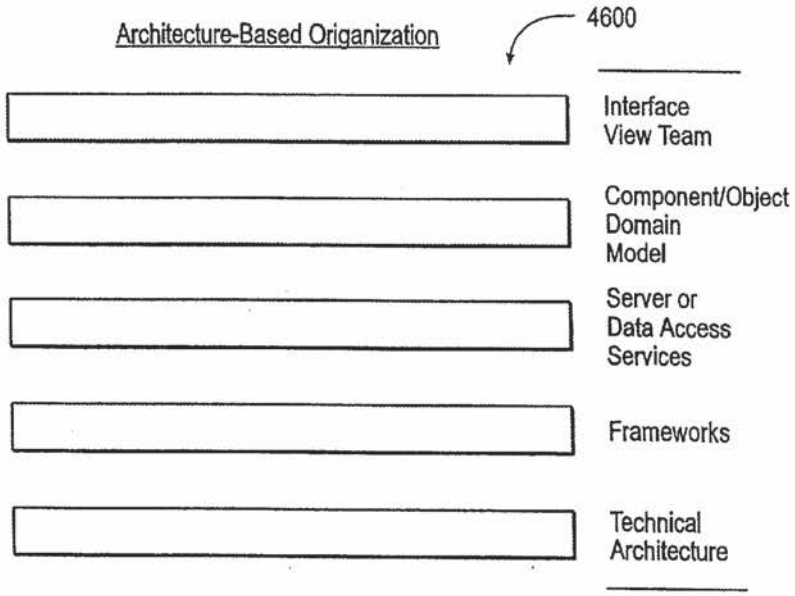

Architecture-Based Origanization
4600
Interface View Team
Component/Object Domain Model
Server or Data Access Services
Frameworks
Technical Architecture

Fig. 46

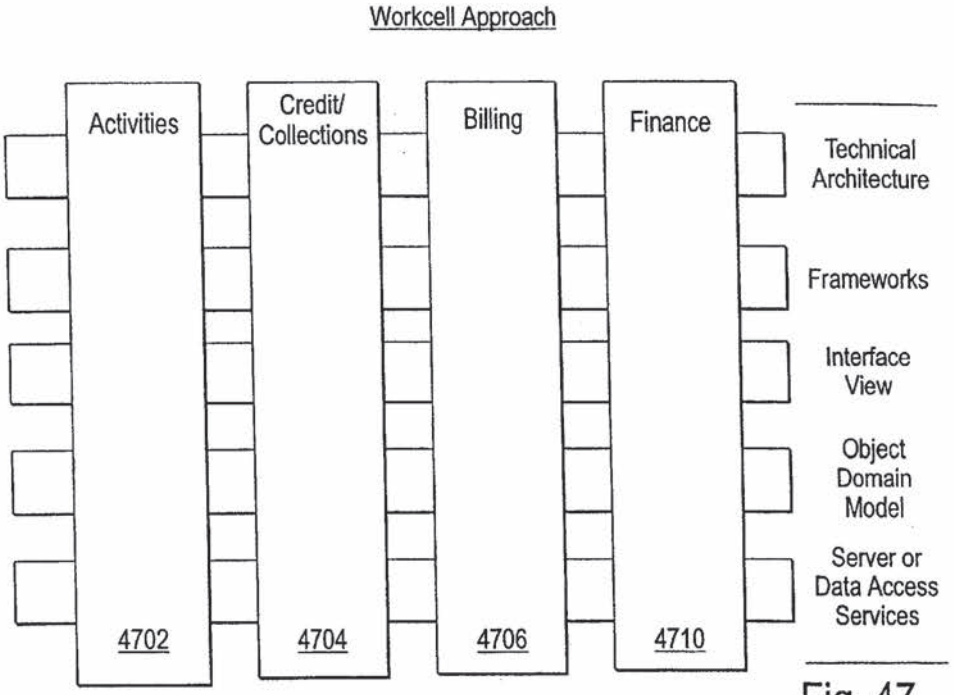

Workcell Approach
Activities
Credit/ Collections
Billing
Finance
Technical Architecture
Frameworks
Interface View
Object Domain Model
Server or Data Access Services
4702
4704
4706
4710

Fig. 47

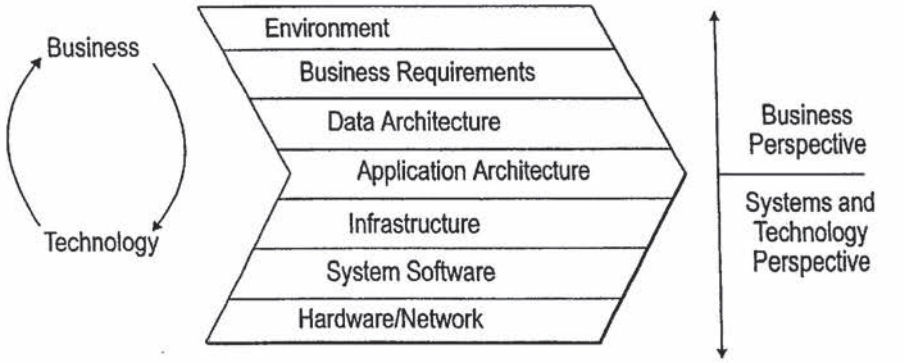

Business
Technology
Environment
Business Requirements
Data Architecture
Application Architecture
Infrastructure
System Software
Hardware/Network
Business Perspective
Systems and Technology Perspective

Fig. 48

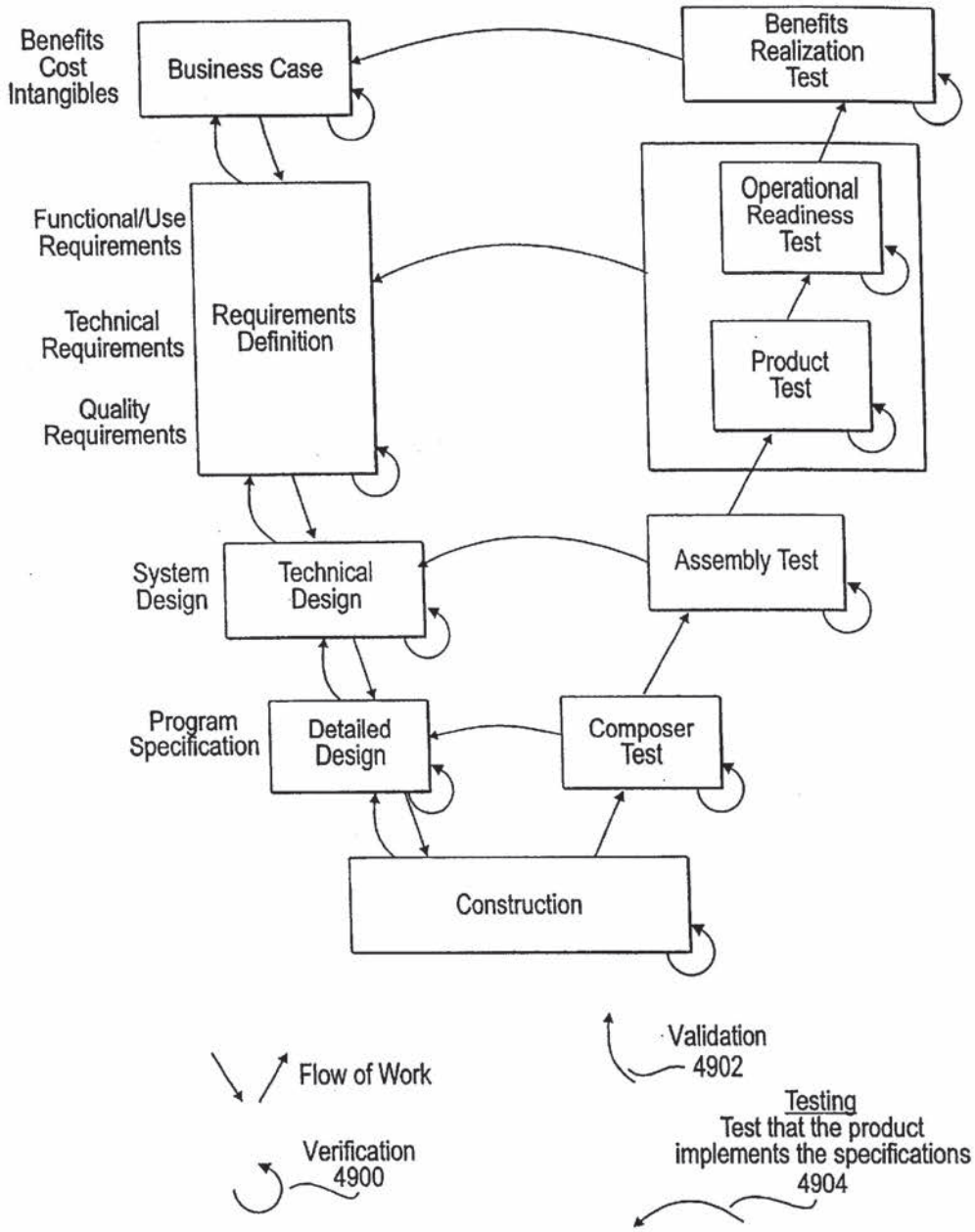

Benefits
Cost
Intangibles
Business Case
Benefits
Realization
Test
Operational
Readiness
Test
Functional/Use
Requirements
Technical
Requirements
Quality
Requirements
Requirements
Definition
Product
Test
System
Design
Technical
Design
Assembly Test
Program
Specification
Detailed
Design
Composer
Test
Construction
Flow of Work
Verification
4900
Validation
4902
Testing
Test that the product
implements the specifications
4904

Fig. 49

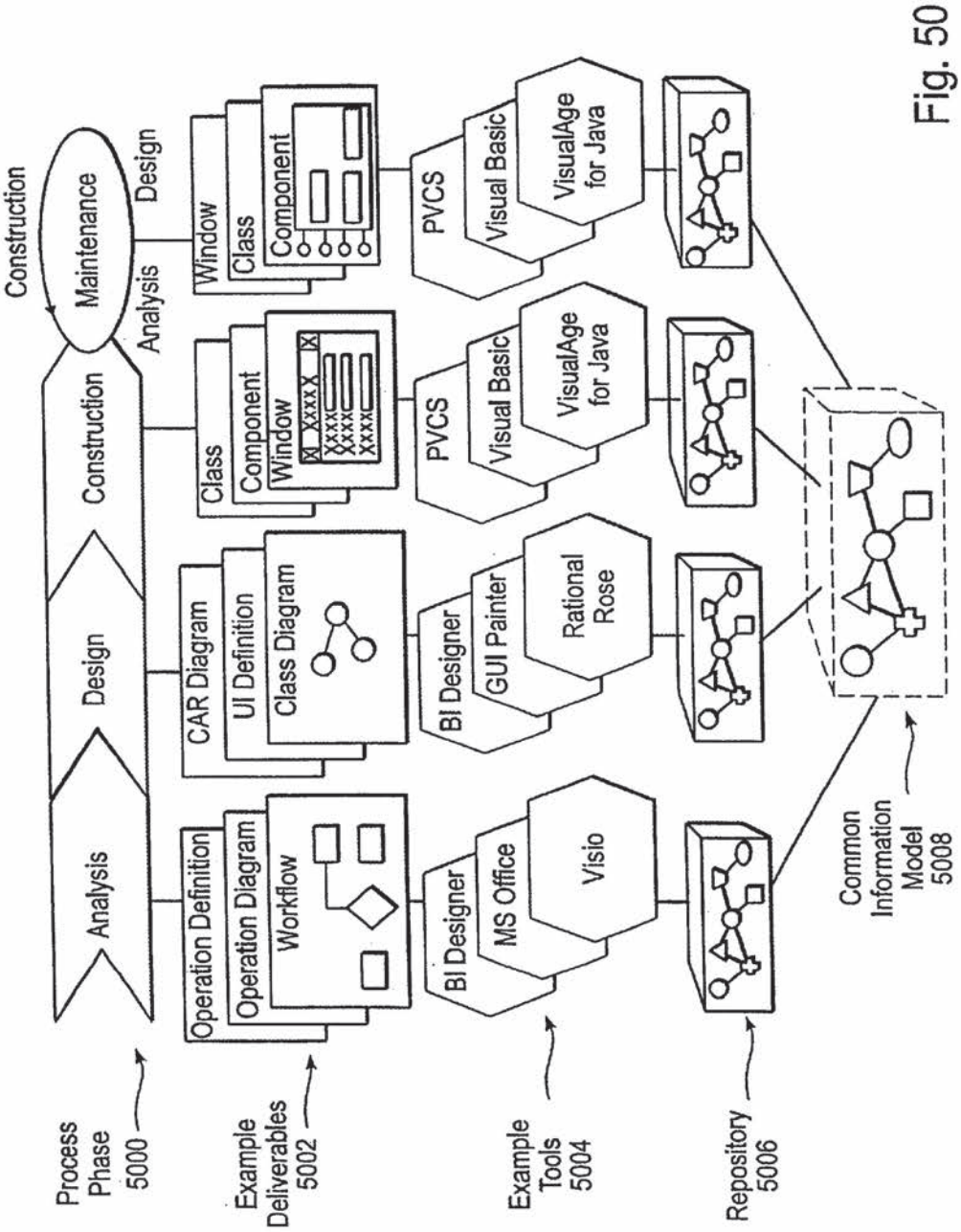

Process Phase 5000
Example Deliverables 5002
Example Tools 5004
Repository 5006
Common Information Model 5008
Analysis
Design
Construction
Maintenance
Construction
Design
Analysis
Operation Definition
Operation Diagram
Workflow
CAR Diagram
UI Definition
Class Diagram
Class
Component
Window
Xxxx
Window
Class
Component
BI Designer
MS Office
Visio
BI Designer
GUI Painter
Rational Rose
PVCS
Visual Basic
VisualAge for Java
PVCS
Visual Basic
VisualAge for Java

Fig. 50

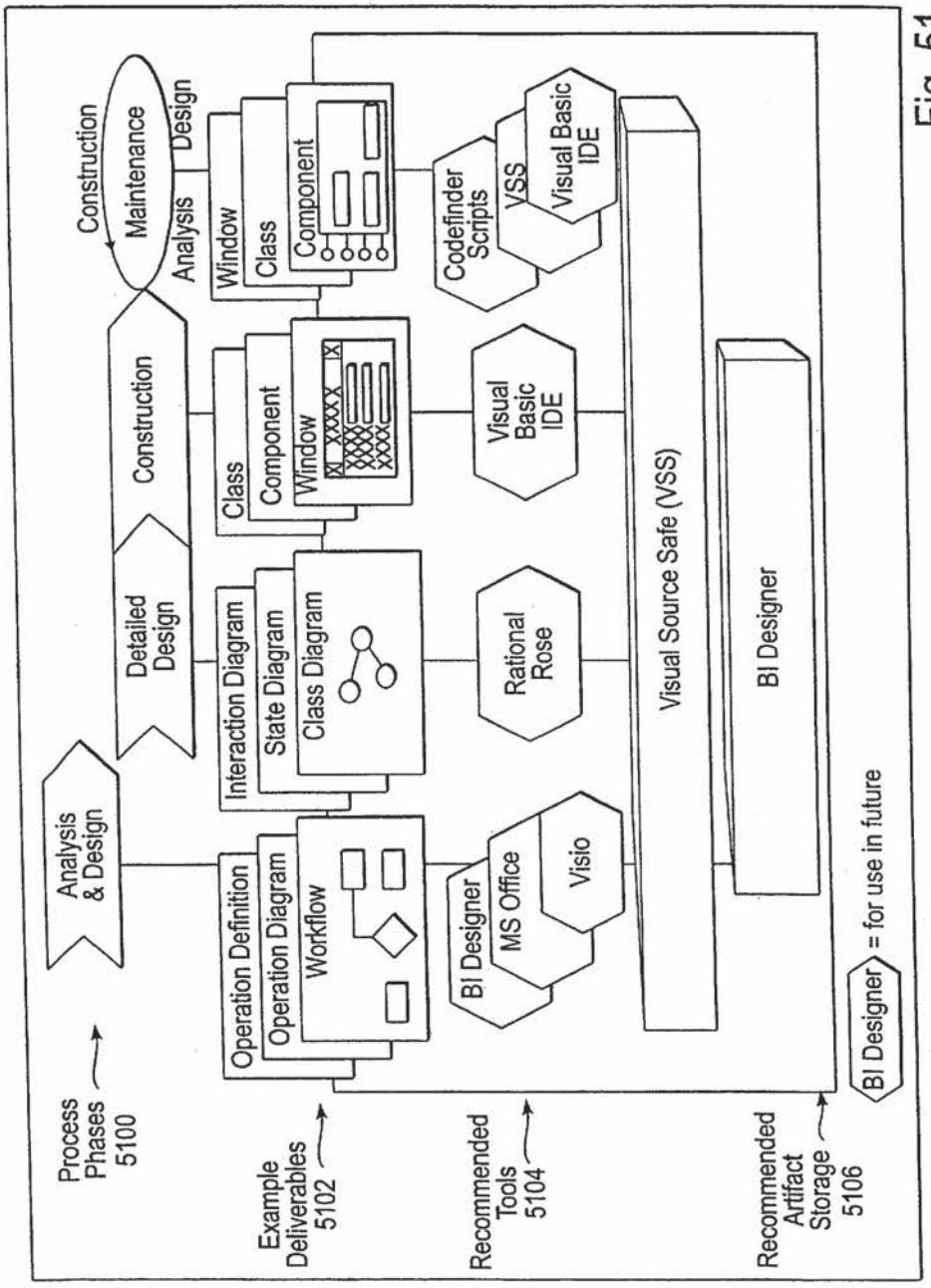

Process Phases 5100
Example Deliverables 5102
Recommended Tools 5104
Recommended Artifact Storage 5106
Analysis & Design
Detailed Design
Construction
Maintenance
Analysis
Design
Construction
Operation Definition
Operation Diagram
Workflow
Interaction Diagram
State Diagram
Class Diagram
Class
Component
Window
Window
Class
Component
BI Designer
MS Office
Visio
Rational Rose
Visual Basic IDE
Codefinder Scripts
VSS
Visual Basic IDE
Visual Source Safe (VSS)
BI Designer
BI Designer = for use in future

Fig. 51

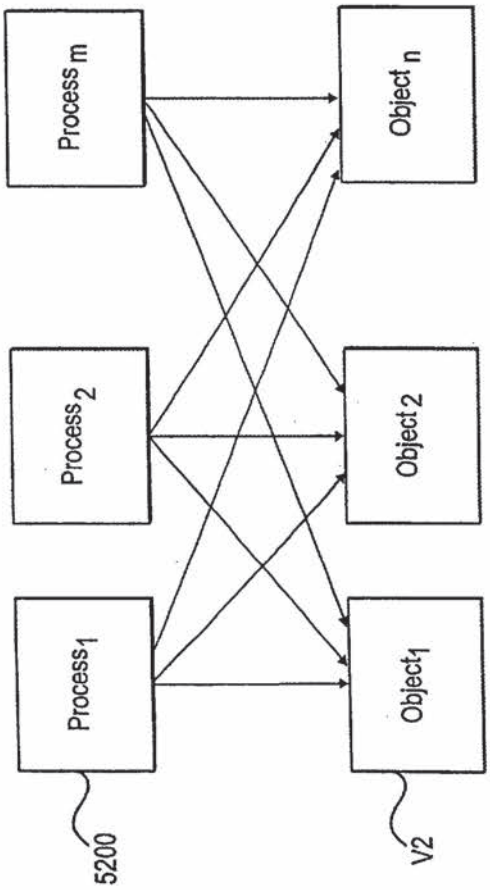

Process m
Process 2
Process 1
Object n
Object 2
Object 1
5200
V2

Fig. 52

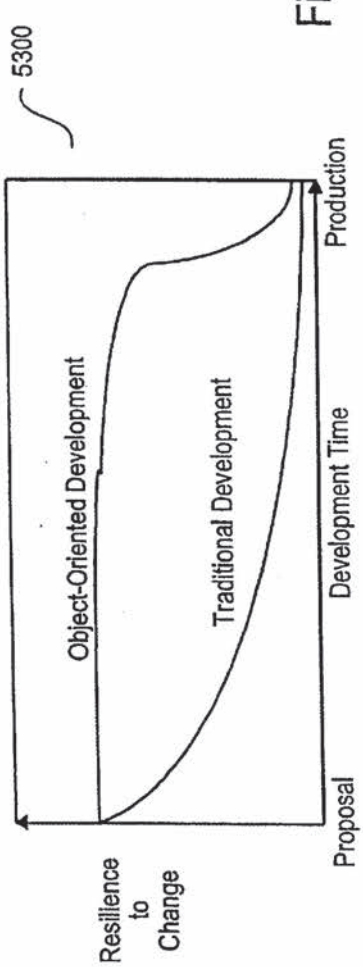

5300
Object-Oriented Development
Traditional Development
Production
Development Time
Proposal
Resilience to Change

Fig. 53

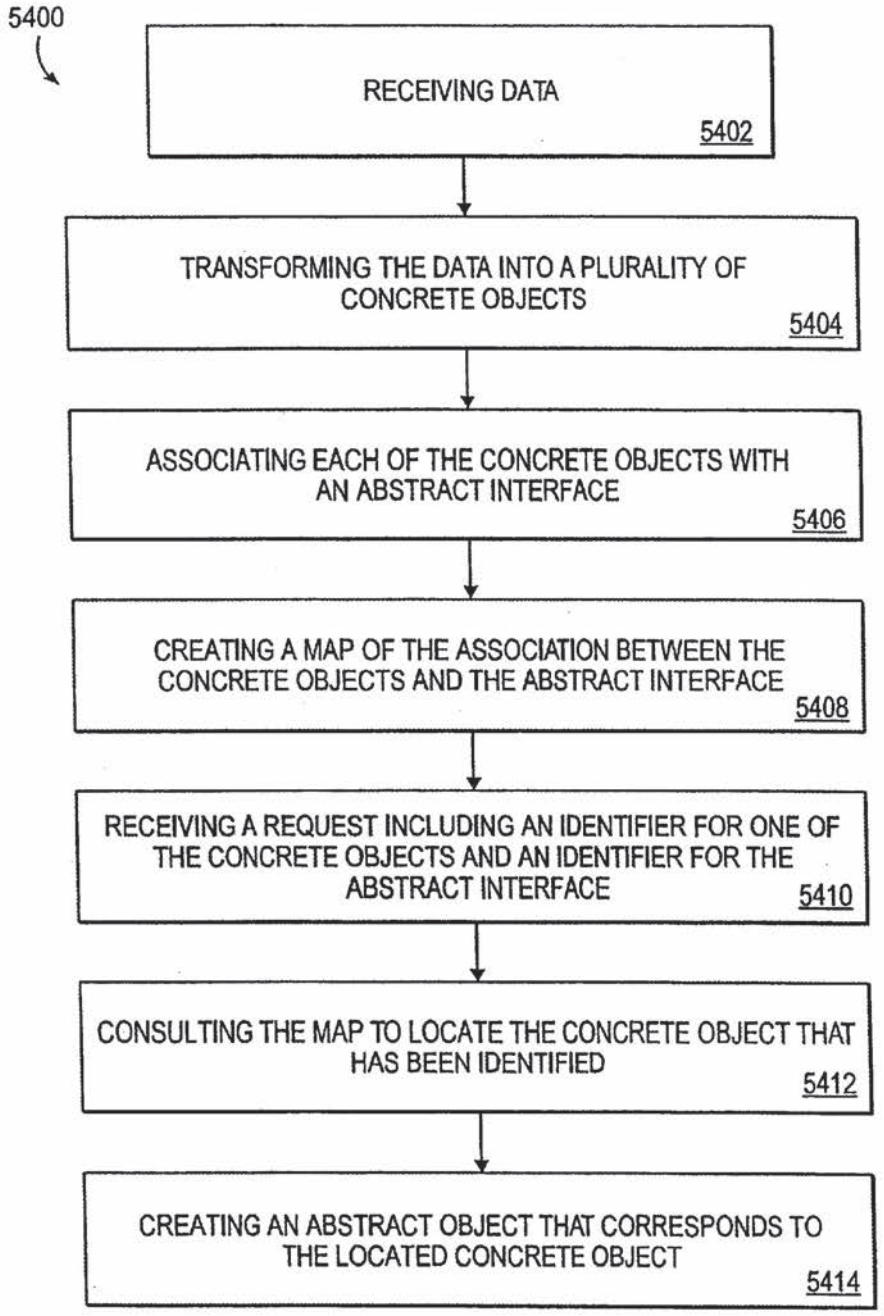

5400
RECEIVING DATA
5402
TRANSFORMING THE DATA INTO A PLURALITY OF CONCRETE OBJECTS
5404
ASSOCIATING EACH OF THE CONCRETE OBJECTS WITH AN ABSTRACT INTERFACE
5406
CREATING A MAP OF THE ASSOCIATION BETWEEN THE CONCRETE OBJECTS AND THE ABSTRACT INTERFACE
5408
RECEIVING A REQUEST INCLUDING AN IDENTIFIER FOR ONE OF THE CONCRETE OBJECTS AND AN IDENTIFIER FOR THE ABSTRACT INTERFACE
5410
CONSULTING THE MAP TO LOCATE THE CONCRETE OBJECT THAT HAS BEEN IDENTIFIED
5412
CREATING AN ABSTRACT OBJECT THAT CORRESPONDS TO THE LOCATED CONCRETE OBJECT
5414

Fig. 54

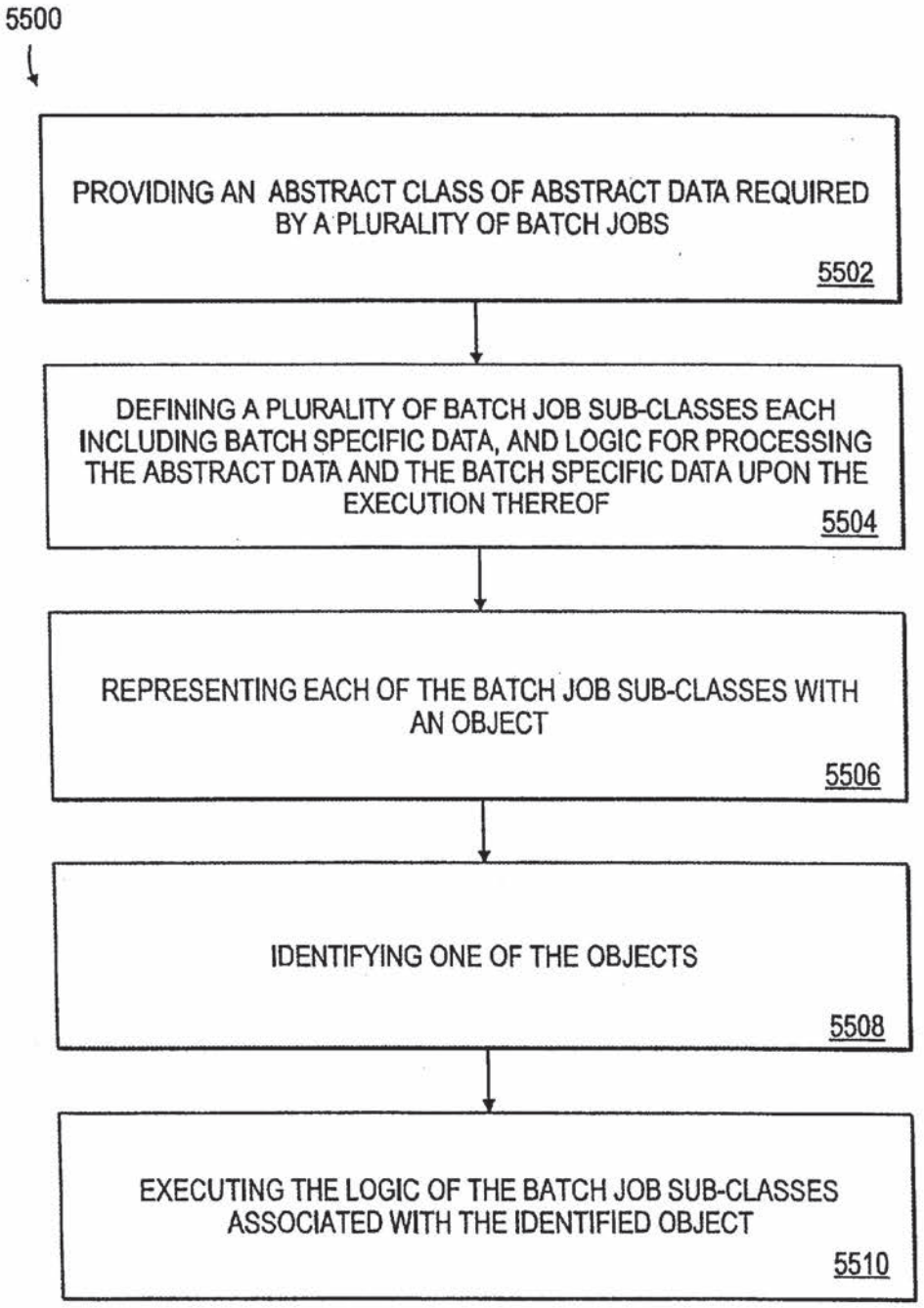

5500
PROVIDING AN ABSTRACT CLASS OF ABSTRACT DATA REQUIRED BY A PLURALITY OF BATCH JOBS
5502
DEFINING A PLURALITY OF BATCH JOB SUB-CLASSES EACH INCLUDING BATCH SPECIFIC DATA, AND LOGIC FOR PROCESSING THE ABSTRACT DATA AND THE BATCH SPECIFIC DATA UPON THE EXECUTION THEREOF
5504
REPRESENTING EACH OF THE BATCH JOB SUB-CLASSES WITH AN OBJECT
5506
IDENTIFYING ONE OF THE OBJECTS
5508
EXECUTING THE LOGIC OF THE BATCH JOB SUB-CLASSES ASSOCIATED WITH THE IDENTIFIED OBJECT
5510

Fig. 55

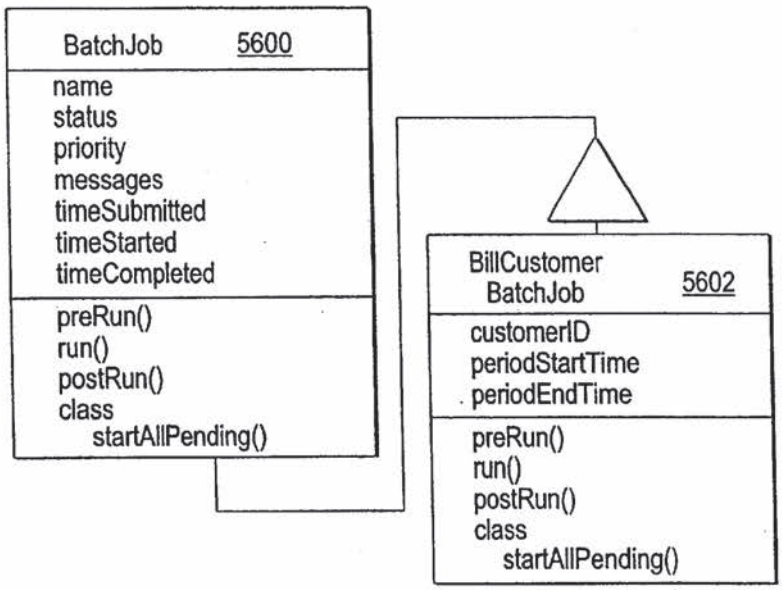

BatchJob 5600
name
status
priority
messages
timeSubmitted
timeStarted
timeCompleted
preRun()
run()
postRun()
class
startAllPending()
BillCustomer
BatchJob 5602
customerID
periodStartTime
periodEndTime
preRun()
run()
postRun()
class
startAllPending()

Fig. 56

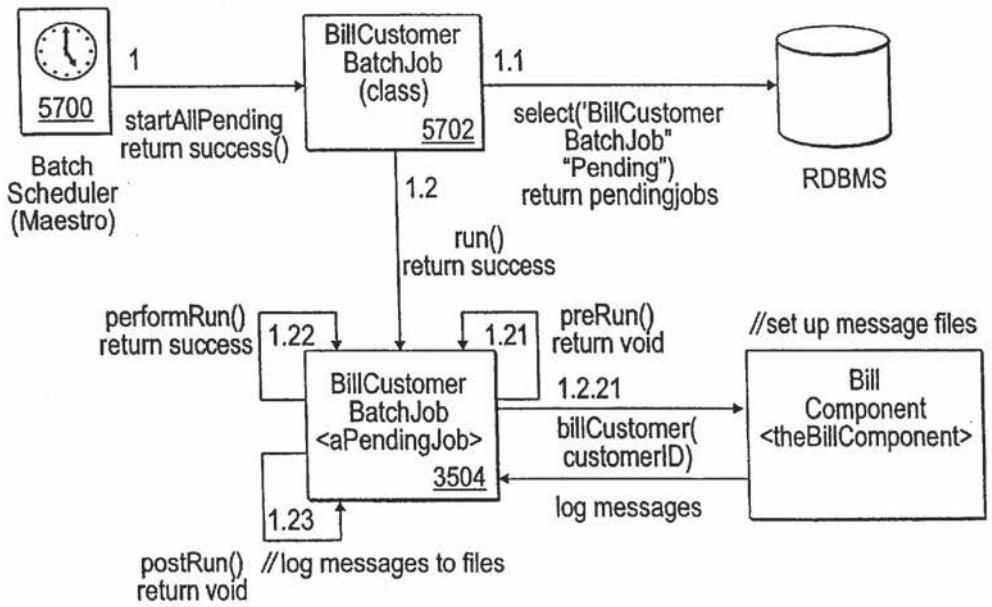

5700
Batch
Scheduler
(Maestro)
1
startAllPending
return success()
BillCustomer
BatchJob
(class)
5702
1.1
select('BillCustomer
BatchJob"
"Pending")
return pendingjobs
RDBMS
1.2
run()
return success
performRun()
return success
1.22
1.21
preRun()
return void
//set up message files
BillCustomer
BatchJob
<aPendingJob>
3504
1.2.21
billCustomer(
customerID)
Bill
Component
<theBillComponent>
log messages
1.23
postRun()
return void
//log messages to files

Fig. 57

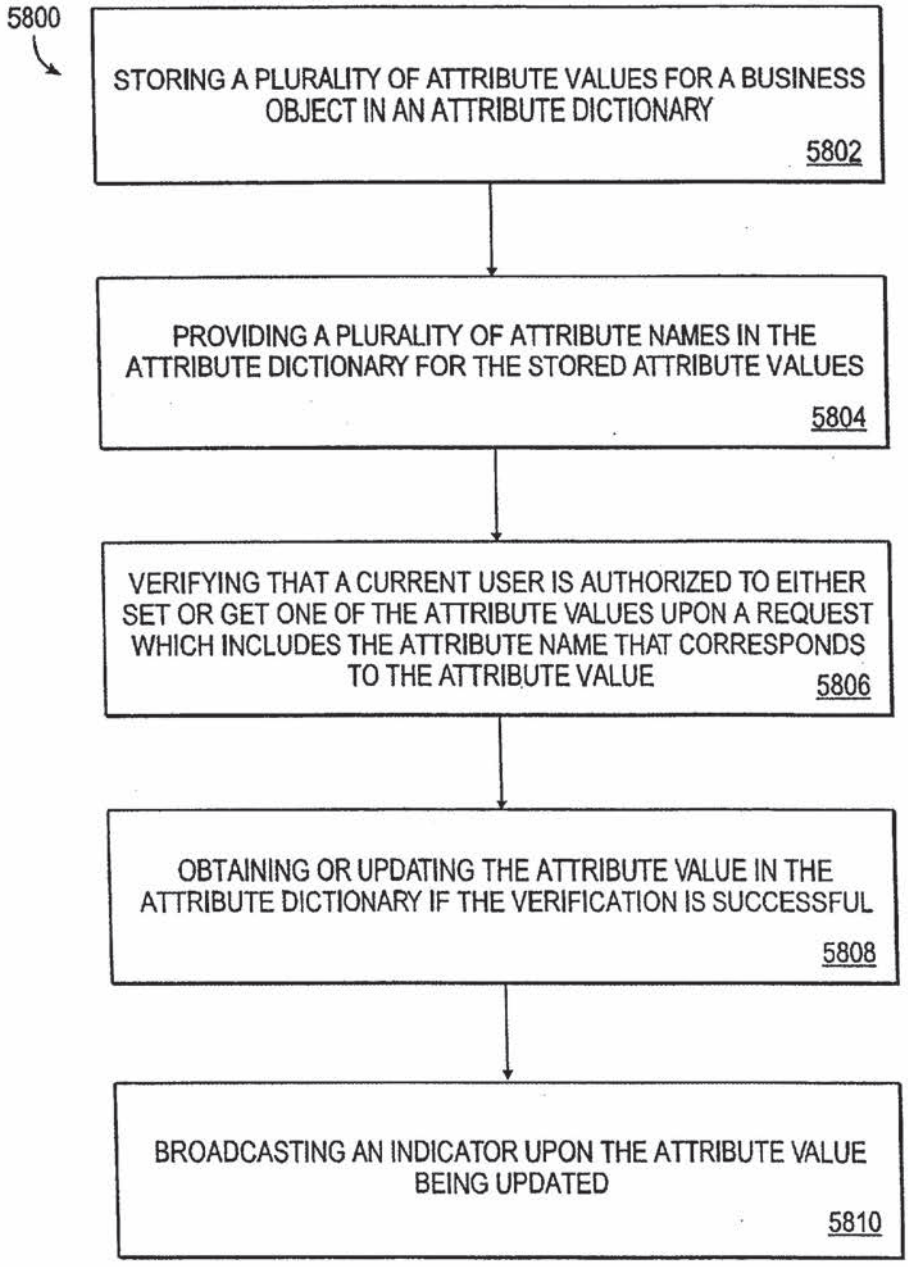

5800
STORING A PLURALITY OF ATTRIBUTE VALUES FOR A BUSINESS OBJECT IN AN ATTRIBUTE DICTIONARY
5802
PROVIDING A PLURALITY OF ATTRIBUTE NAMES IN THE ATTRIBUTE DICTIONARY FOR THE STORED ATTRIBUTE VALUES
5804
VERIFYING THAT A CURRENT USER IS AUTHORIZED TO EITHER SET OR GET ONE OF THE ATTRIBUTE VALUES UPON A REQUEST WHICH INCLUDES THE ATTRIBUTE NAME THAT CORRESPONDS TO THE ATTRIBUTE VALUE
5806
OBTAINING OR UPDATING THE ATTRIBUTE VALUE IN THE ATTRIBUTE DICTIONARY IF THE VERIFICATION IS SUCCESSFUL
5808
BROADCASTING AN INDICATOR UPON THE ATTRIBUTE VALUE BEING UPDATED
5810

Fig. 58

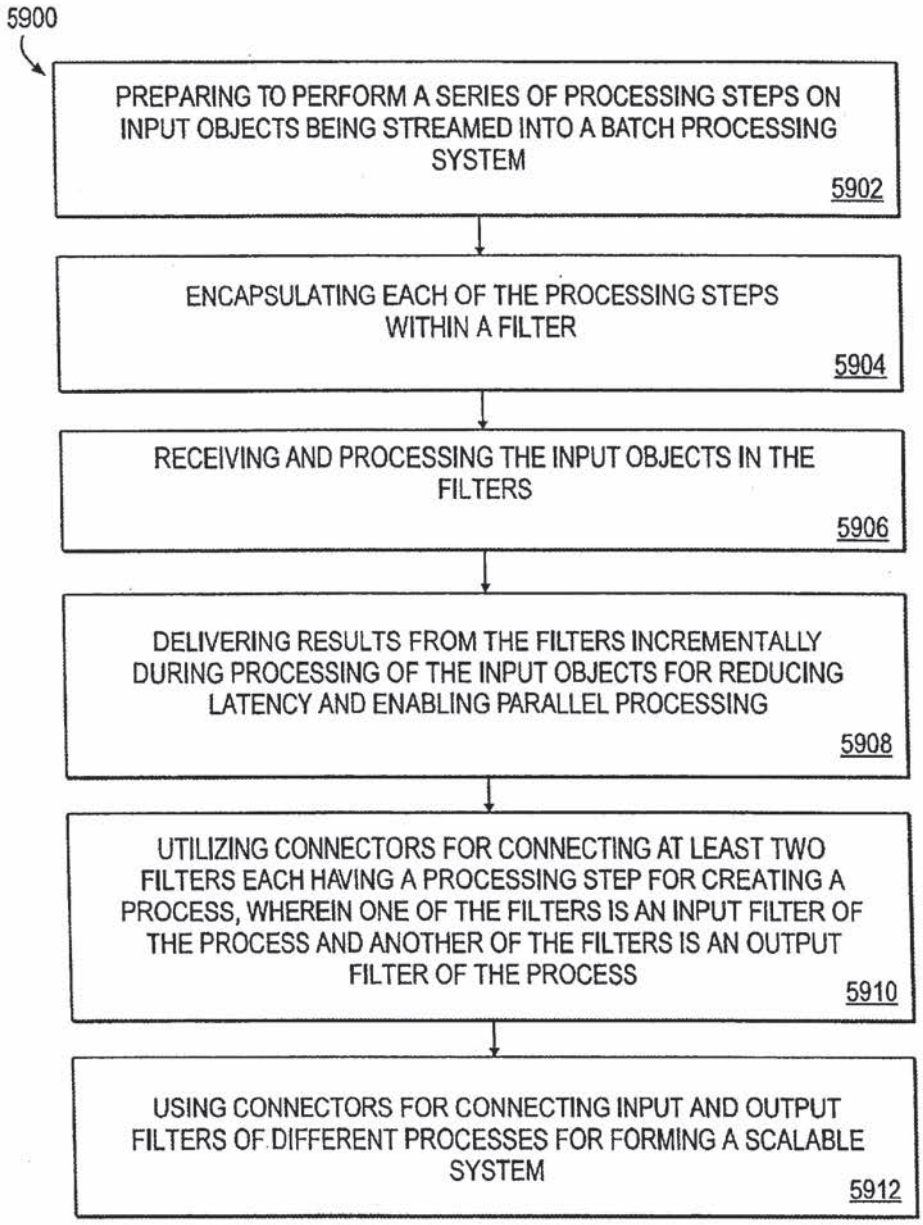

5900
PREPARING TO PERFORM A SERIES OF PROCESSING STEPS ON INPUT OBJECTS BEING STREAMED INTO A BATCH PROCESSING SYSTEM
5902
ENCAPSULATING EACH OF THE PROCESSING STEPS WITHIN A FILTER
5904
RECEIVING AND PROCESSING THE INPUT OBJECTS IN THE FILTERS
5906
DELIVERING RESULTS FROM THE FILTERS INCREMENTALLY DURING PROCESSING OF THE INPUT OBJECTS FOR REDUCING LATENCY AND ENABLING PARALLEL PROCESSING
5908
UTILIZING CONNECTORS FOR CONNECTING AT LEAST TWO FILTERS EACH HAVING A PROCESSING STEP FOR CREATING A PROCESS, WHEREIN ONE OF THE FILTERS IS AN INPUT FILTER OF THE PROCESS AND ANOTHER OF THE FILTERS IS AN OUTPUT FILTER OF THE PROCESS
5910
USING CONNECTORS FOR CONNECTING INPUT AND OUTPUT FILTERS OF DIFFERENT PROCESSES FOR FORMING A SCALABLE SYSTEM
5912

Fig. 59

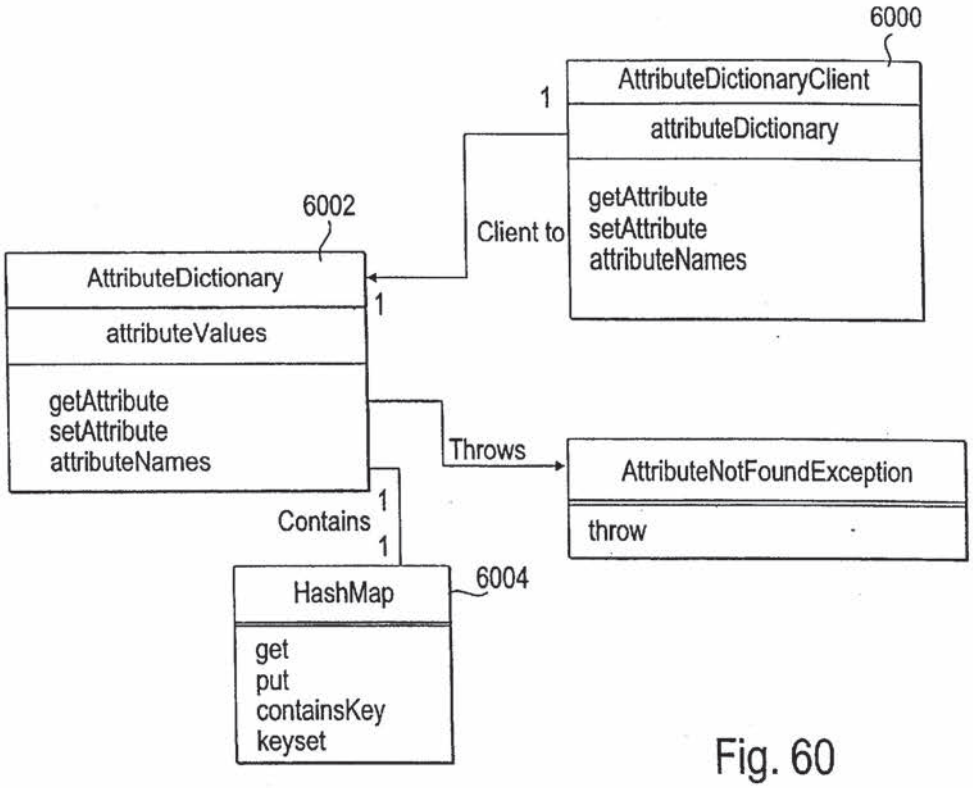

6000
AttributeDictionaryClient
attributeDictionary
getAttribute
setAttribute
attributeNames
1
Client to
6002
AttributeDictionary
attributeValues
getAttribute
setAttribute
attributeNames
1
Throws
AttributeNotFoundException
throw
Contains
1
1
HashMap
6004
get
put
containsKey
keyset

Fig. 60

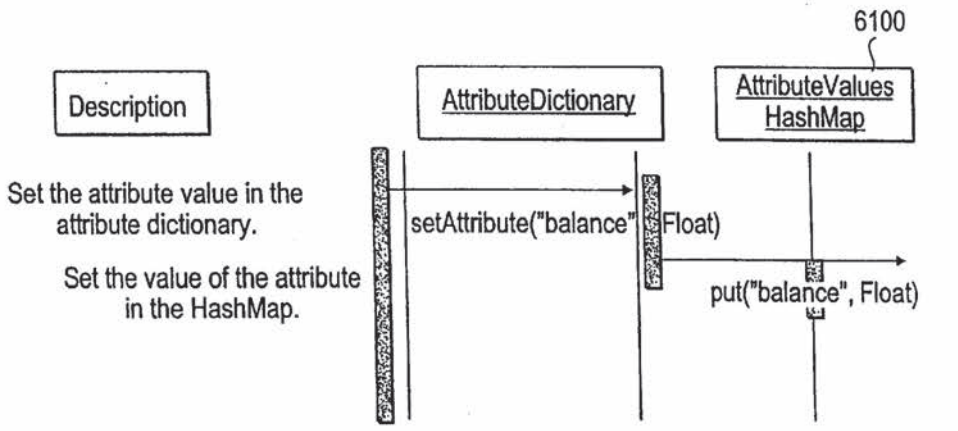

6100
Description
AttributeDictionary
AttributeValues
HashMap
Set the attribute value in the attribute dictionary.
setAttribute("balance", Float)
Set the value of the attribute in the HashMap.
put("balance", Float)

Fig. 61

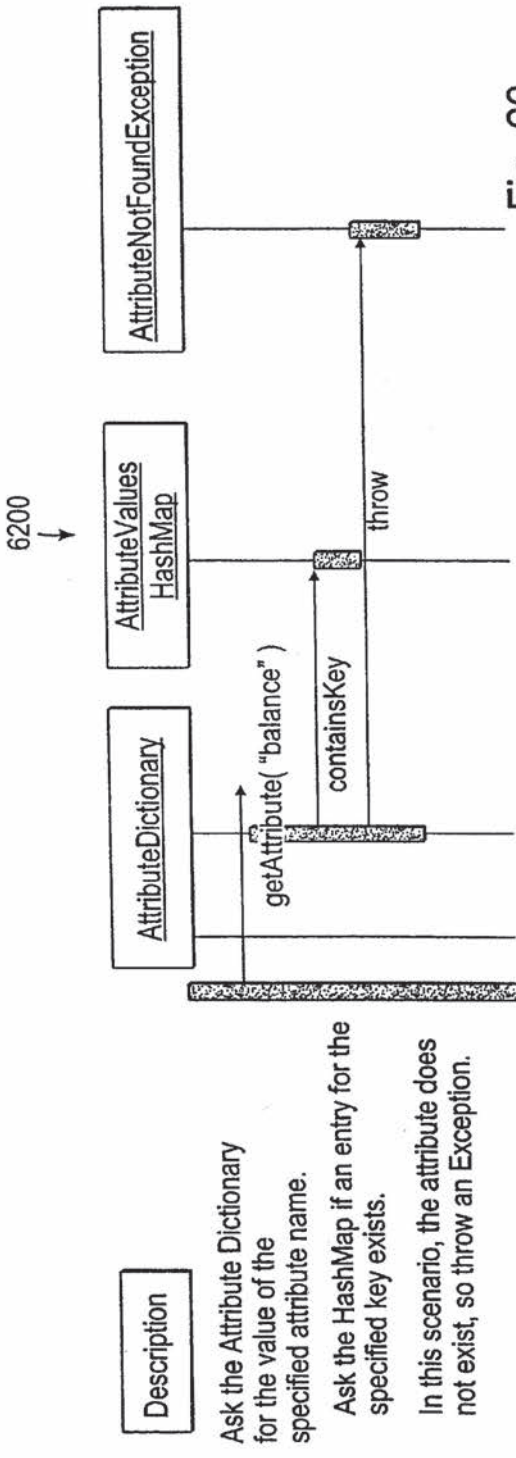

6200
AttributeDictionary
AttributeValues HashMap
AttributeNotFoundException
getAttribute("balance")
containsKey
throw
Description
Ask the Attribute Dictionary for the value of the specified attribute name.
Ask the HashMap if an entry for the specified key exists.
In this scenario, the attribute does not exist, so throw an Exception.

Fig. 62

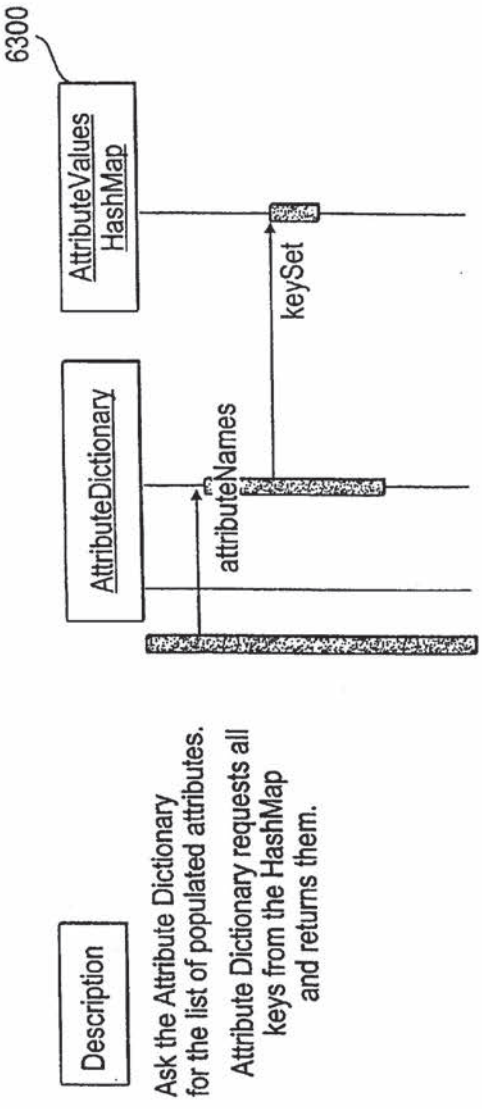

6300
AttributeDictionary
AttributeValues HashMap
attributeNames
keySet
Description
Ask the Attribute Dictionary for the list of populated attributes.
Attribute Dictionary requests all keys from the HashMap and returns them.

Fig. 63

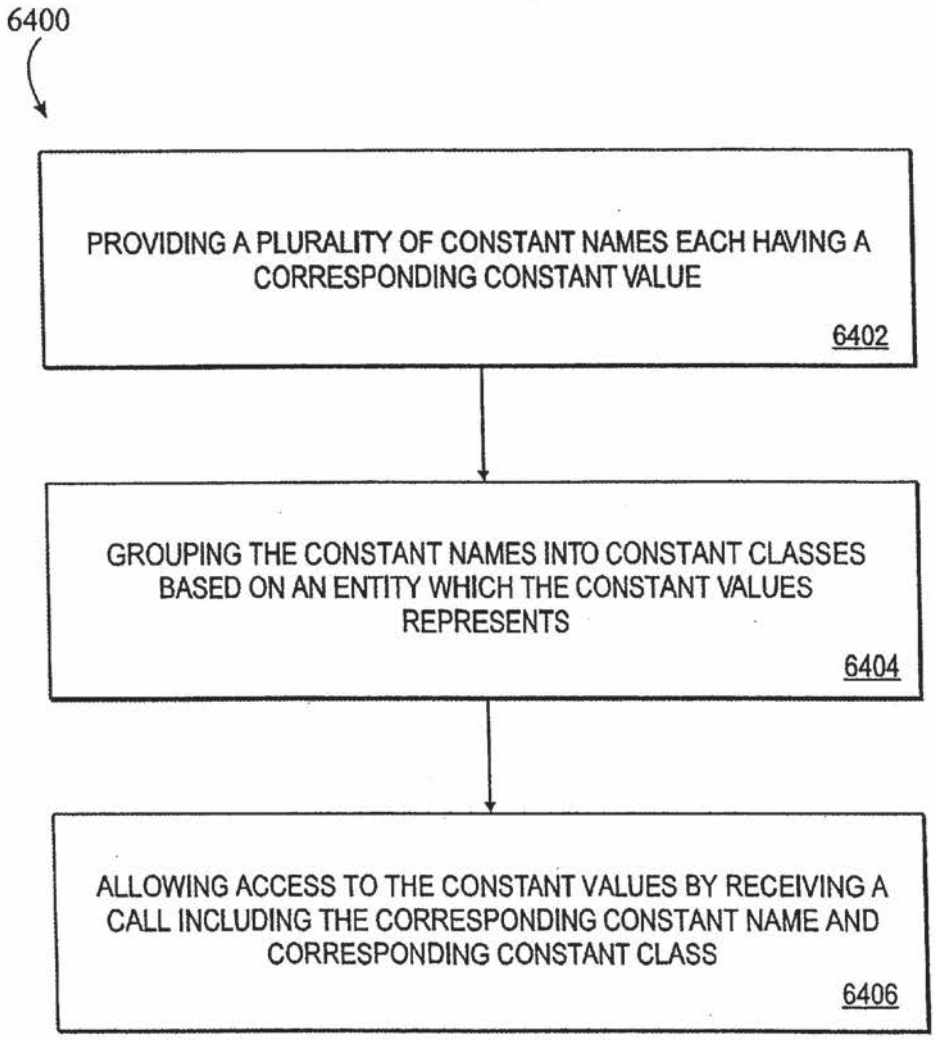

6400
PROVIDING A PLURALITY OF CONSTANT NAMES EACH HAVING A CORRESPONDING CONSTANT VALUE
6402
GROUPING THE CONSTANT NAMES INTO CONSTANT CLASSES BASED ON AN ENTITY WHICH THE CONSTANT VALUES REPRESENTS
6404
ALLOWING ACCESS TO THE CONSTANT VALUES BY RECEIVING A CALL INCLUDING THE CORRESPONDING CONSTANT NAME AND CORRESPONDING CONSTANT CLASS
6406

Fig. 64

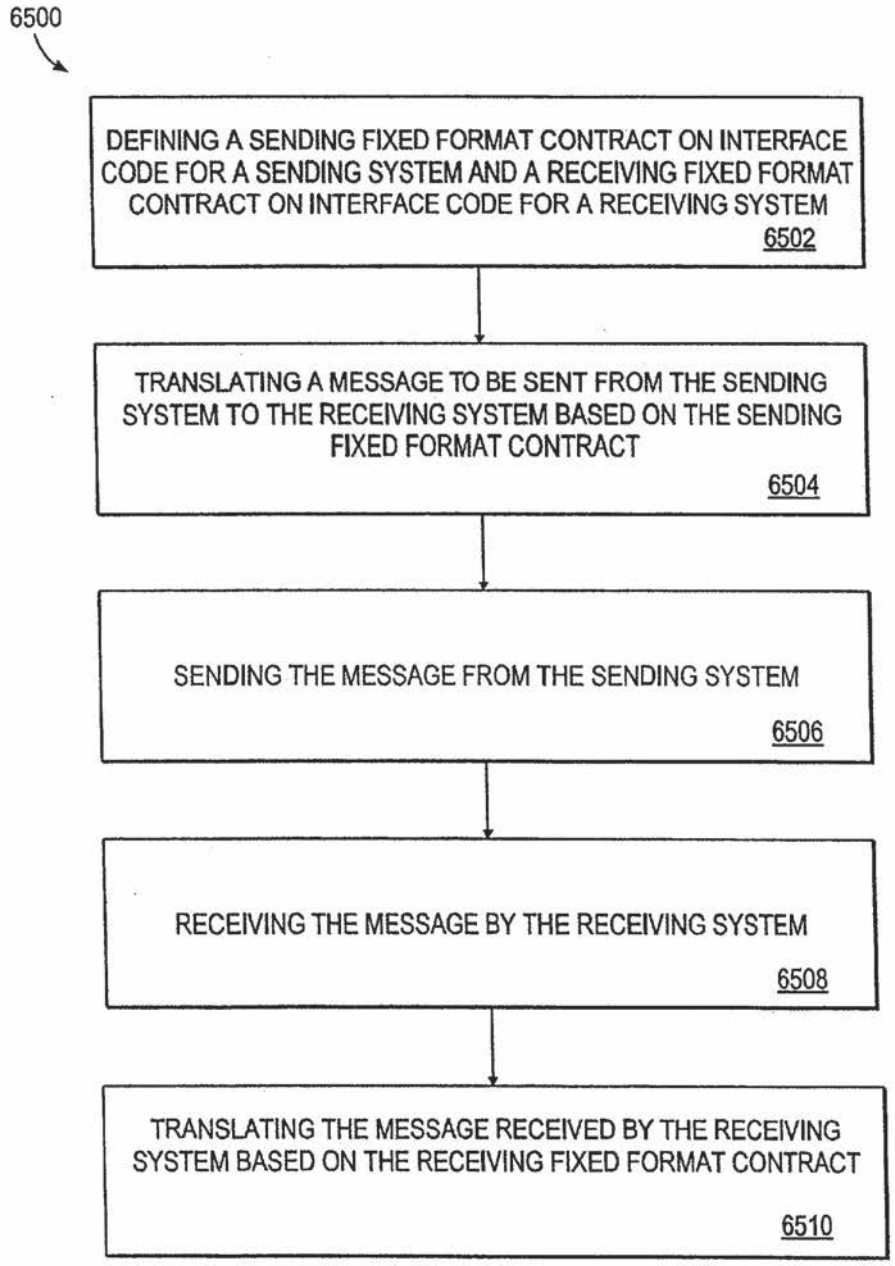

6500
DEFINING A SENDING FIXED FORMAT CONTRACT ON INTERFACE CODE FOR A SENDING SYSTEM AND A RECEIVING FIXED FORMAT CONTRACT ON INTERFACE CODE FOR A RECEIVING SYSTEM
6502
TRANSLATING A MESSAGE TO BE SENT FROM THE SENDING SYSTEM TO THE RECEIVING SYSTEM BASED ON THE SENDING FIXED FORMAT CONTRACT
6504
SENDING THE MESSAGE FROM THE SENDING SYSTEM
6506
RECEIVING THE MESSAGE BY THE RECEIVING SYSTEM
6508
TRANSLATING THE MESSAGE RECEIVED BY THE RECEIVING SYSTEM BASED ON THE RECEIVING FIXED FORMAT CONTRACT
6510

Fig. 65

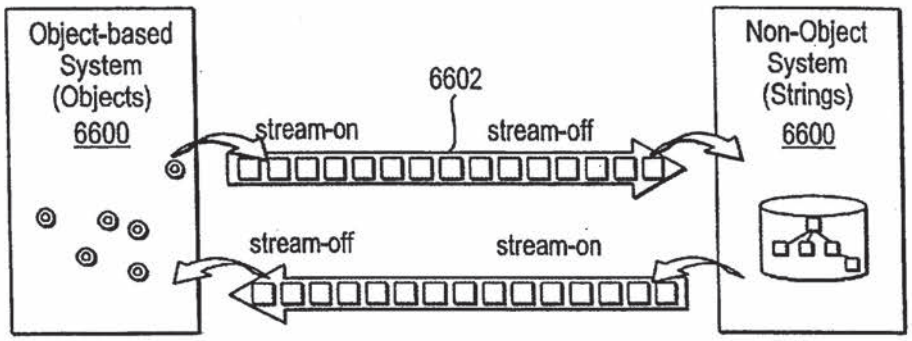

Object-based System (Objects) 6600
6602
stream-on
stream-off
Non-Object System (Strings) 6600
stream-off
stream-on

Fig. 66

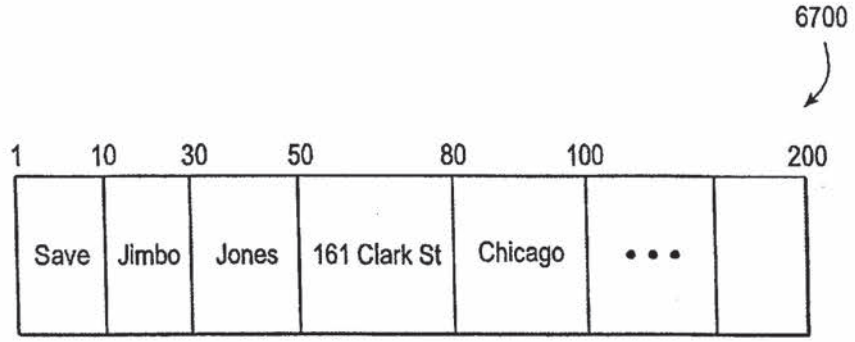

6700
1
10
30
50
80
100
200
Save
Jimbo
Jones
161 Clark St
Chicago
• • •

Fig. 67

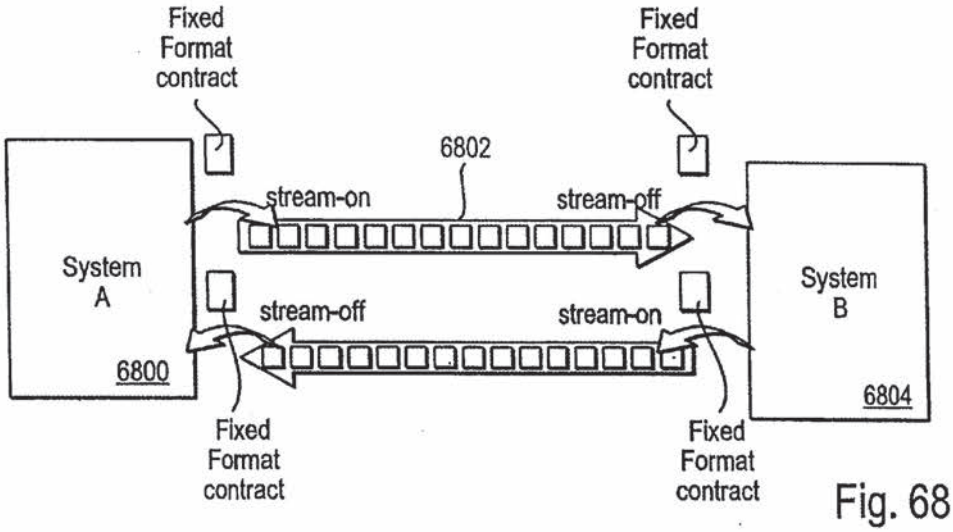

Fixed Format contract
Fixed Format contract
6802
stream-on
stream-off
System A
6800
stream-off
stream-on
System B
6804
Fixed Format contract
Fixed Format contract

Fig. 68

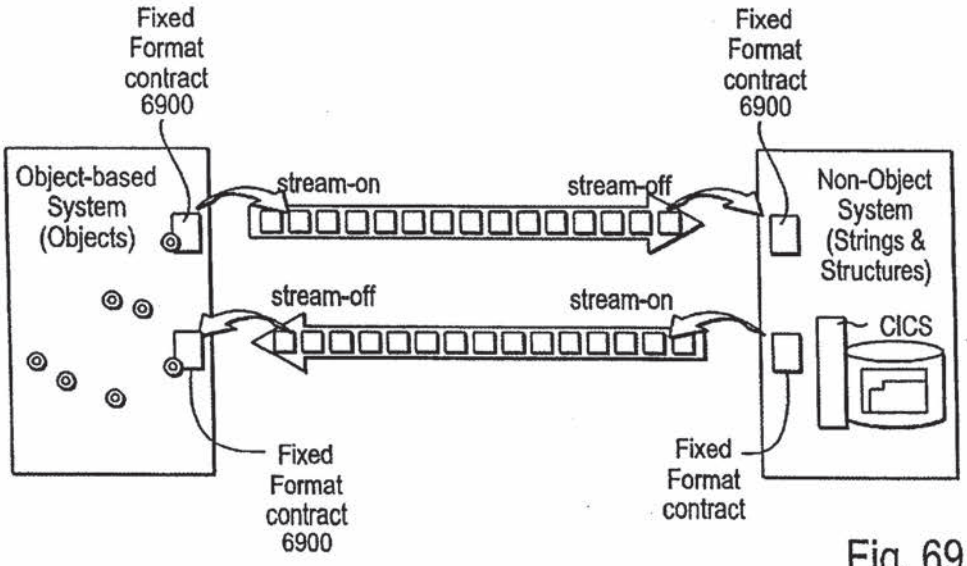

Fixed Format contract 6900
Fixed Format contract 6900
Object-based System (Objects)
stream-on
stream-off
Non-Object System (Strings & Structures)
stream-off
stream-on
CICS
Fixed Format contract 6900
Fixed Format contract

Fig. 69

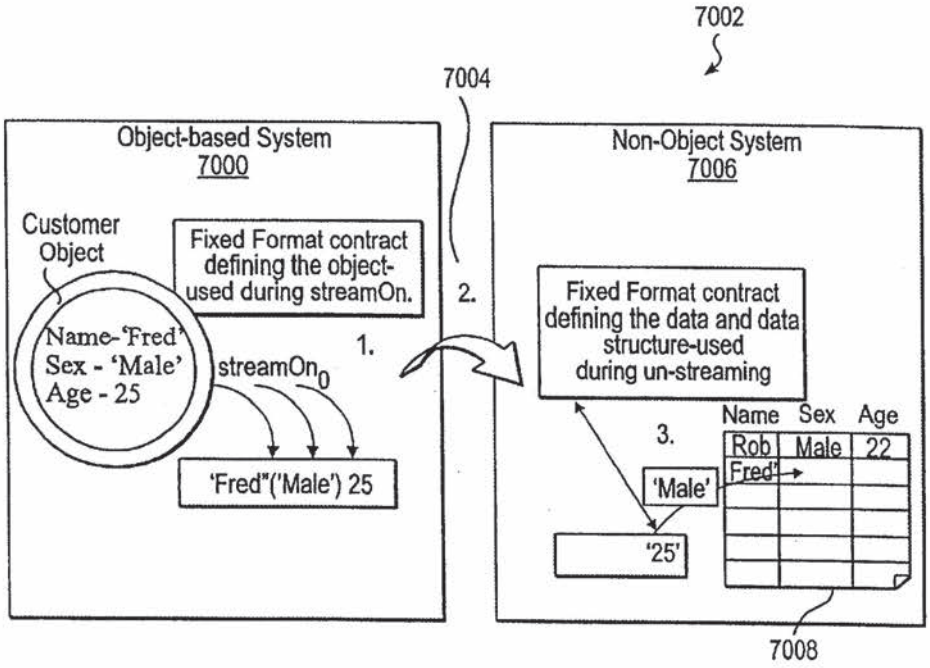

7002
7004
Object-based System
7000
Customer Object
Name-'Fred'
Sex - 'Male'
Age - 25
Fixed Format contract defining the object-used during streamOn.
streamOn0
1.
'Fred''Male') 25
2.
Non-Object System
7006
Fixed Format contract defining the data and data structure-used during un-streaming
3.
Name Sex Age
Rob Male 22
Fred
'Male'
'25'
7008

Fig. 70

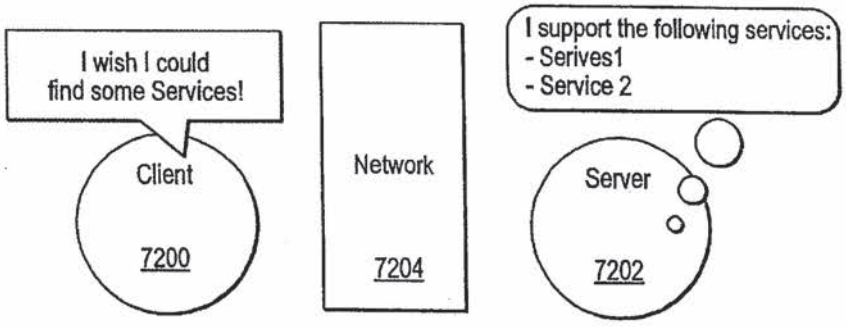

I wish I could find some Services!
Client
7200
Network
7204
I support the following services:
- Serives1
- Service 2
Server
7202

Fig. 72

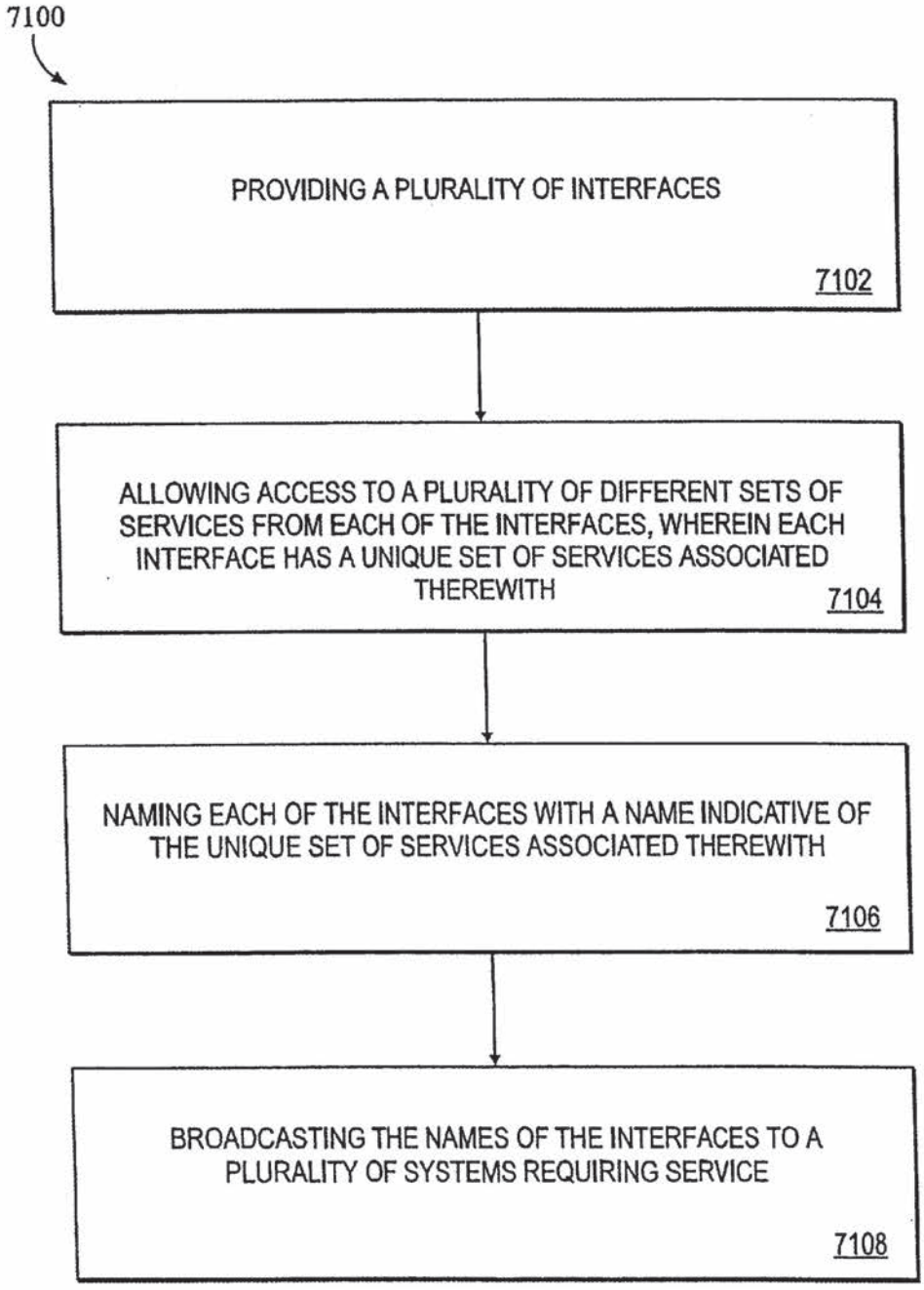

7100
PROVIDING A PLURALITY OF INTERFACES
7102
ALLOWING ACCESS TO A PLURALITY OF DIFFERENT SETS OF SERVICES FROM EACH OF THE INTERFACES, WHEREIN EACH INTERFACE HAS A UNIQUE SET OF SERVICES ASSOCIATED THEREWITH
7104
NAMING EACH OF THE INTERFACES WITH A NAME INDICATIVE OF THE UNIQUE SET OF SERVICES ASSOCIATED THEREWITH
7106
BROADCASTING THE NAMES OF THE INTERFACES TO A PLURALITY OF SYSTEMS REQUIRING SERVICE
7108

Fig. 71

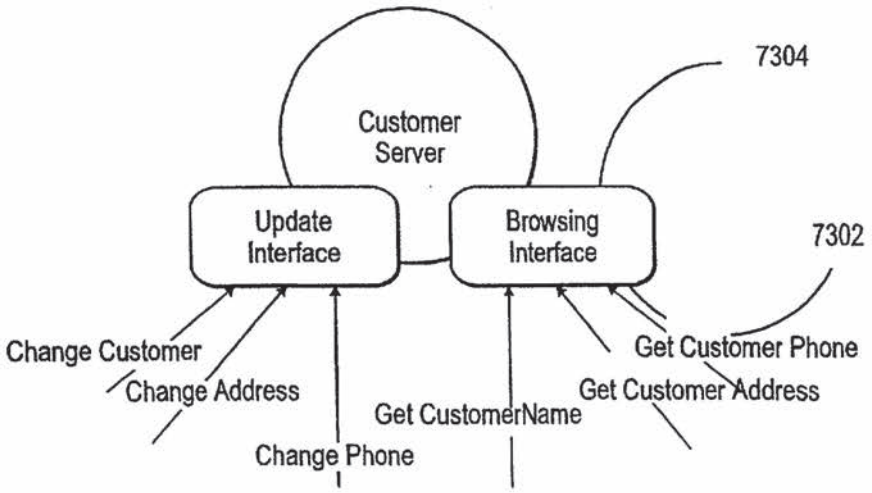

7304
Customer Server
Update Interface
Browsing Interface
7302
Change Customer
Change Address
Change Phone
Get CustomerName
Get Customer Phone
Get Customer Address

Fig. 73

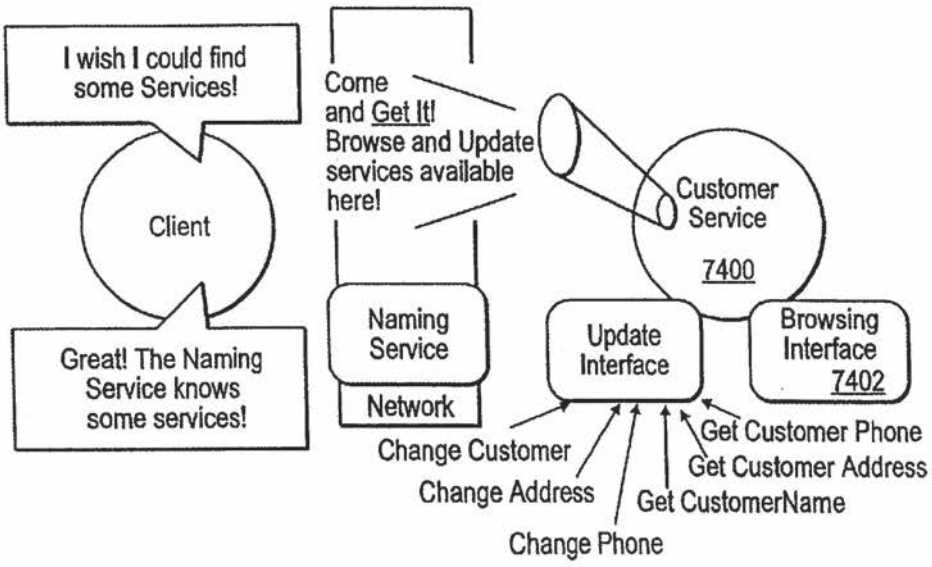

I wish I could find some Services!
Client
Great! The Naming Service knows some services!
Come and Get It! Browse and Update services available here!
Naming Service
Network
Customer Service
7400
Update Interface
Browsing Interface
7402
Change Customer
Change Address
Change Phone
Get CustomerName
Get Customer Phone
Get Customer Address

Fig. 74

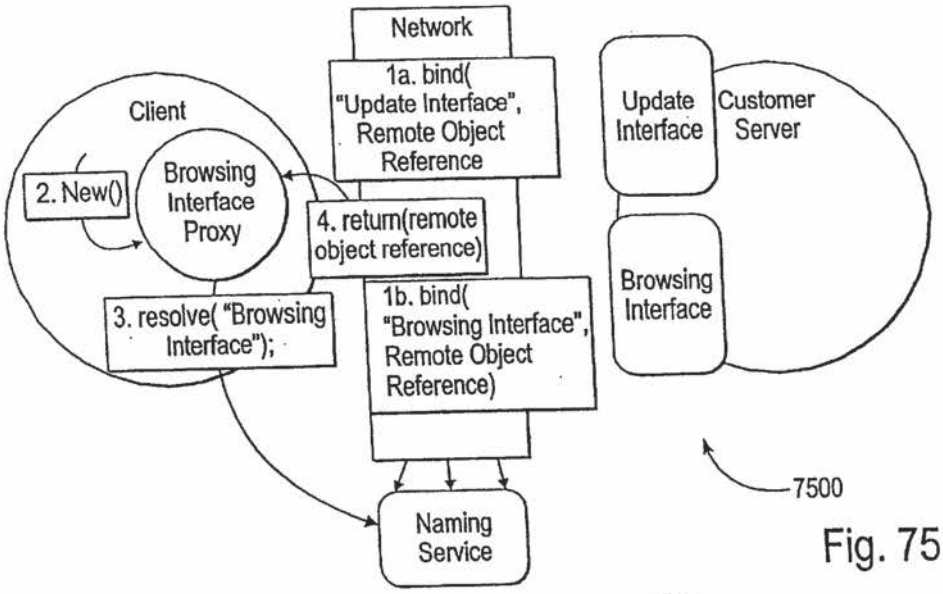

Network
1a. bind("Update Interface", Remote Object Reference
Client
Browsing Interface Proxy
2. New()
4. return(remote object reference)
3. resolve("Browsing Interface");
1b. bind("Browsing Interface", Remote Object Reference)
Update Interface
Customer Server
Browsing Interface
7500
Naming Service

Fig. 75

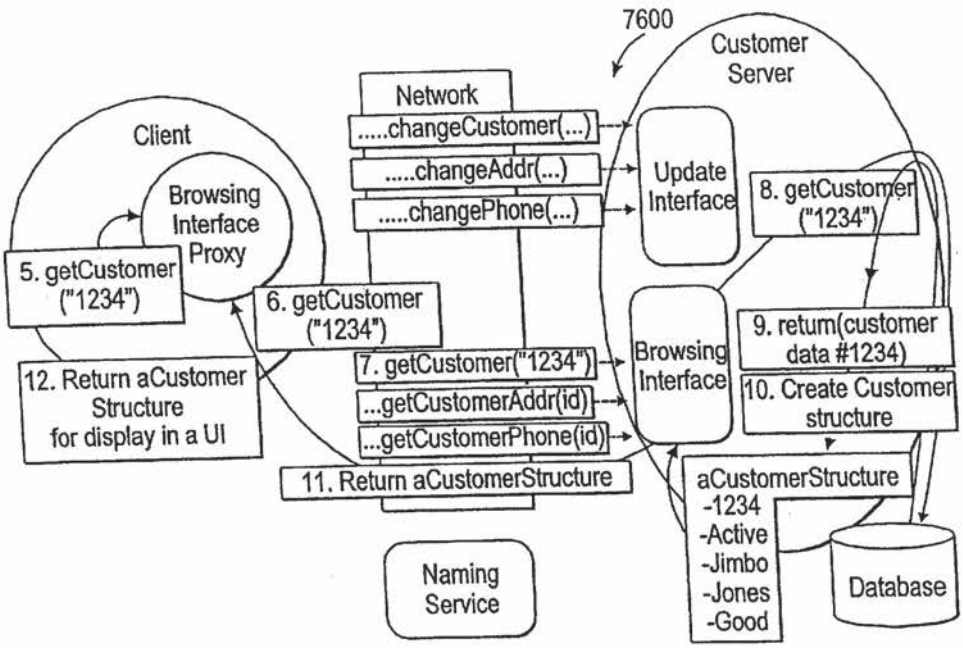

7600
Customer Server
Network
.....changeCustomer(...)
.....changeAddr(...)
.....changePhone(...)
Client
Browsing Interface Proxy
5. getCustomer ("1234")
6. getCustomer ("1234")
12. Return aCustomer Structure for display in a UI
7. getCustomer("1234")
...getCustomerAddr(id)
...getCustomerPhone(id)
11. Return aCustomerStructure
Update Interface
Browsing Interface
8. getCustomer ("1234")
9. return(customer data #1234)
10. Create Customer structure
aCustomerStructure
-1234
-Active
-Jimbo
-Jones
-Good
Database
Naming Service

Fig. 76

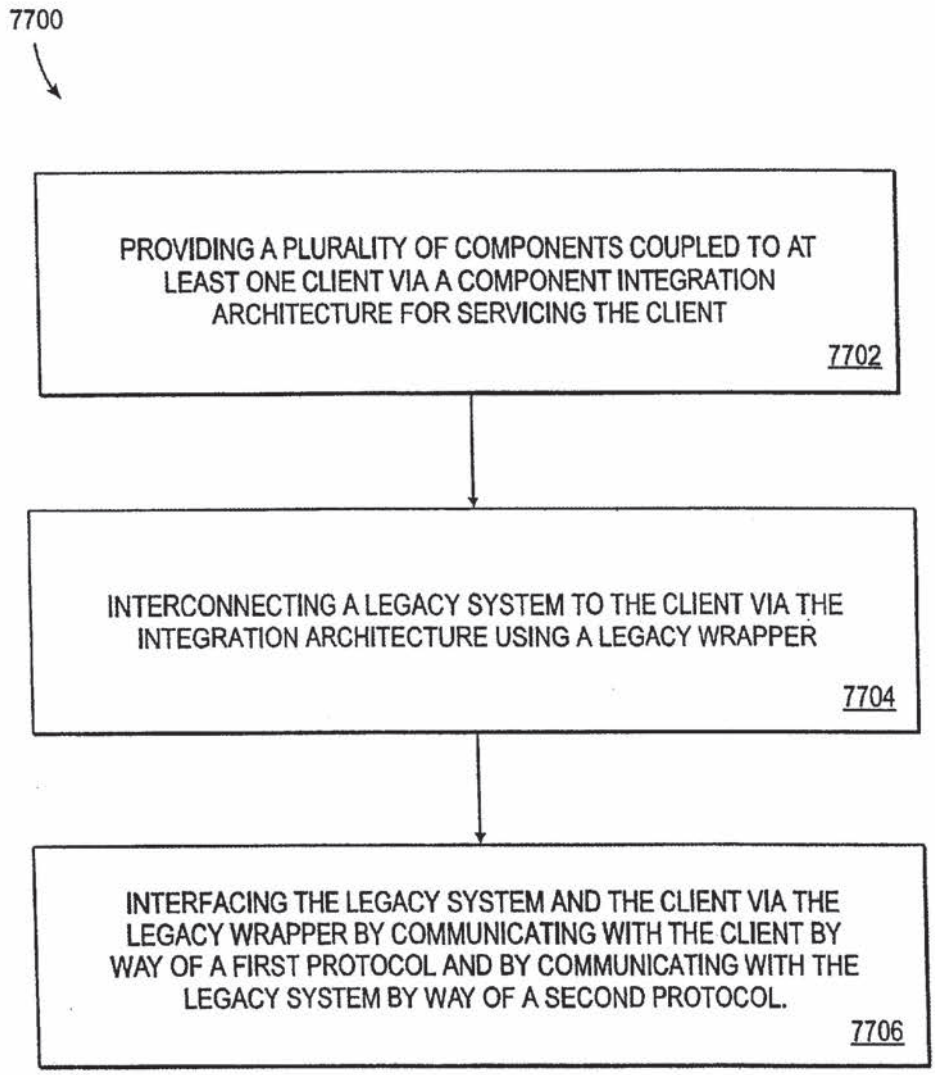

7700
PROVIDING A PLURALITY OF COMPONENTS COUPLED TO AT LEAST ONE CLIENT VIA A COMPONENT INTEGRATION ARCHITECTURE FOR SERVICING THE CLIENT
7702
INTERCONNECTING A LEGACY SYSTEM TO THE CLIENT VIA THE INTEGRATION ARCHITECTURE USING A LEGACY WRAPPER
7704
INTERFACING THE LEGACY SYSTEM AND THE CLIENT VIA THE LEGACY WRAPPER BY COMMUNICATING WITH THE CLIENT BY WAY OF A FIRST PROTOCOL AND BY COMMUNICATING WITH THE LEGACY SYSTEM BY WAY OF A SECOND PROTOCOL.
7706

Fig. 77

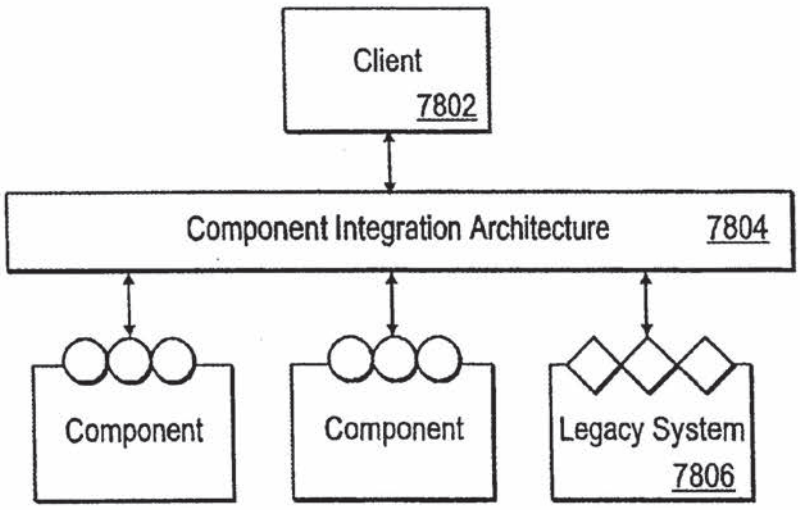

Client
7802
Component Integration Architecture
7804
Component
Component
Legacy System
7806

Fig. 78

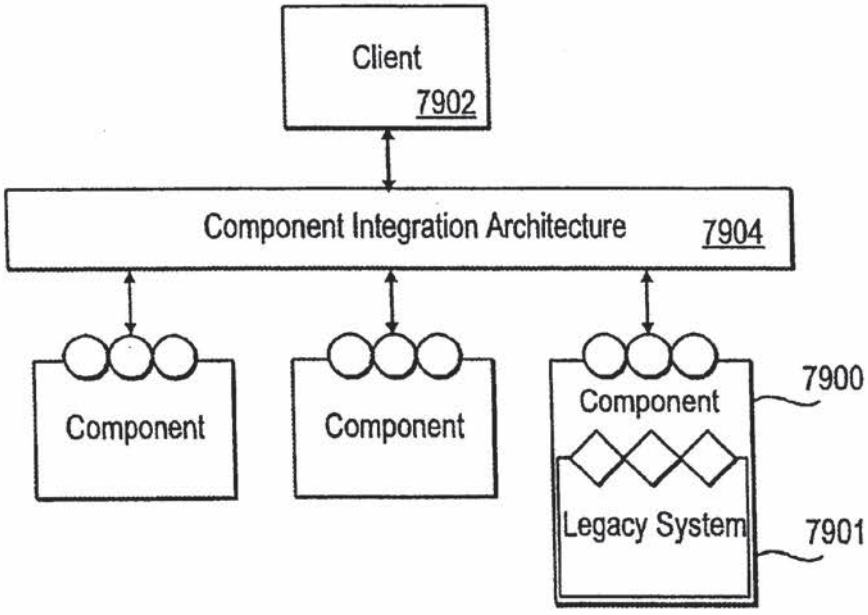

Client
7902
Component Integration Architecture
7904
Component
Component
Component
7900
Legacy System
7901

Fig. 79

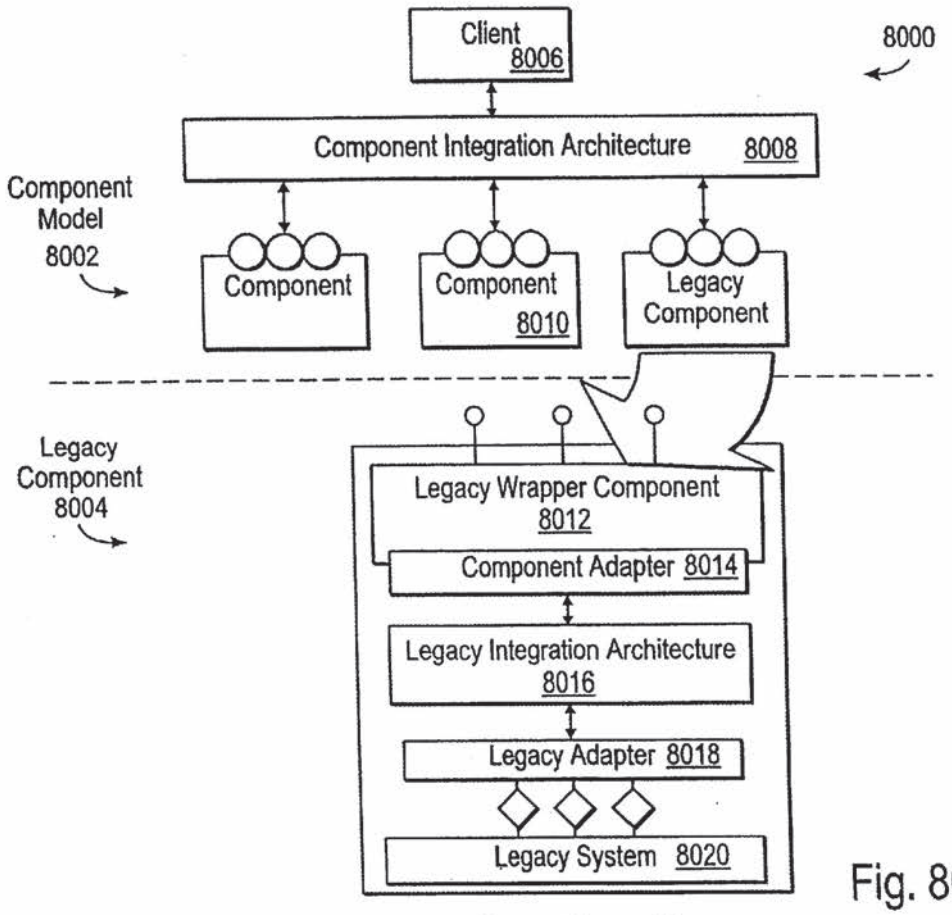

Client
8006
8000
Component Integration Architecture 8008
Component Model 8002
Component
Component
8010
Legacy Component
Legacy Component 8004
Legacy Wrapper Component 8012
Component Adapter 8014
Legacy Integration Architecture 8016
Legacy Adapter 8018
Legacy System 8020

Fig. 80

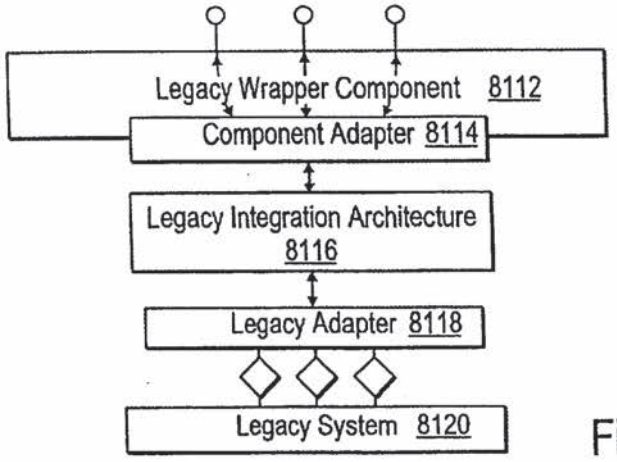

Legacy Wrapper Component 8112
Component Adapter 8114
Legacy Integration Architecture 8116
Legacy Adapter 8118
Legacy System 8120

Fig. 81

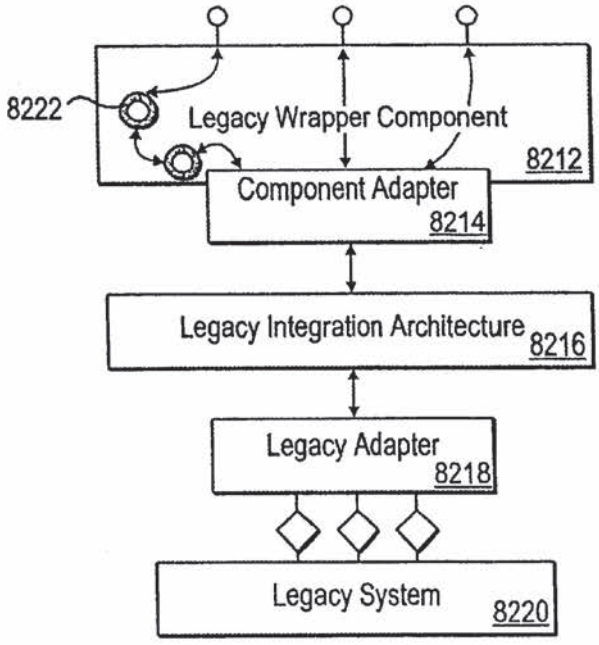

8222
Legacy Wrapper Component
8212
Component Adapter
8214
Legacy Integration Architecture
8216
Legacy Adapter
8218
Legacy System
8220

Fig. 82

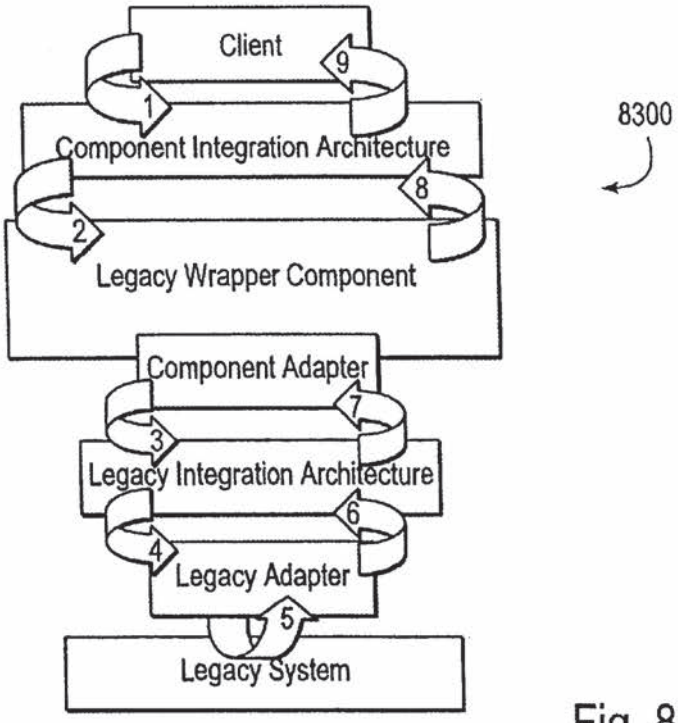

Client
9
1
8300
Component Integration Architecture
8
2
Legacy Wrapper Component
Component Adapter
7
3
Legacy Integration Architecture
6
4
Legacy Adapter
5
Legacy System

Fig. 83

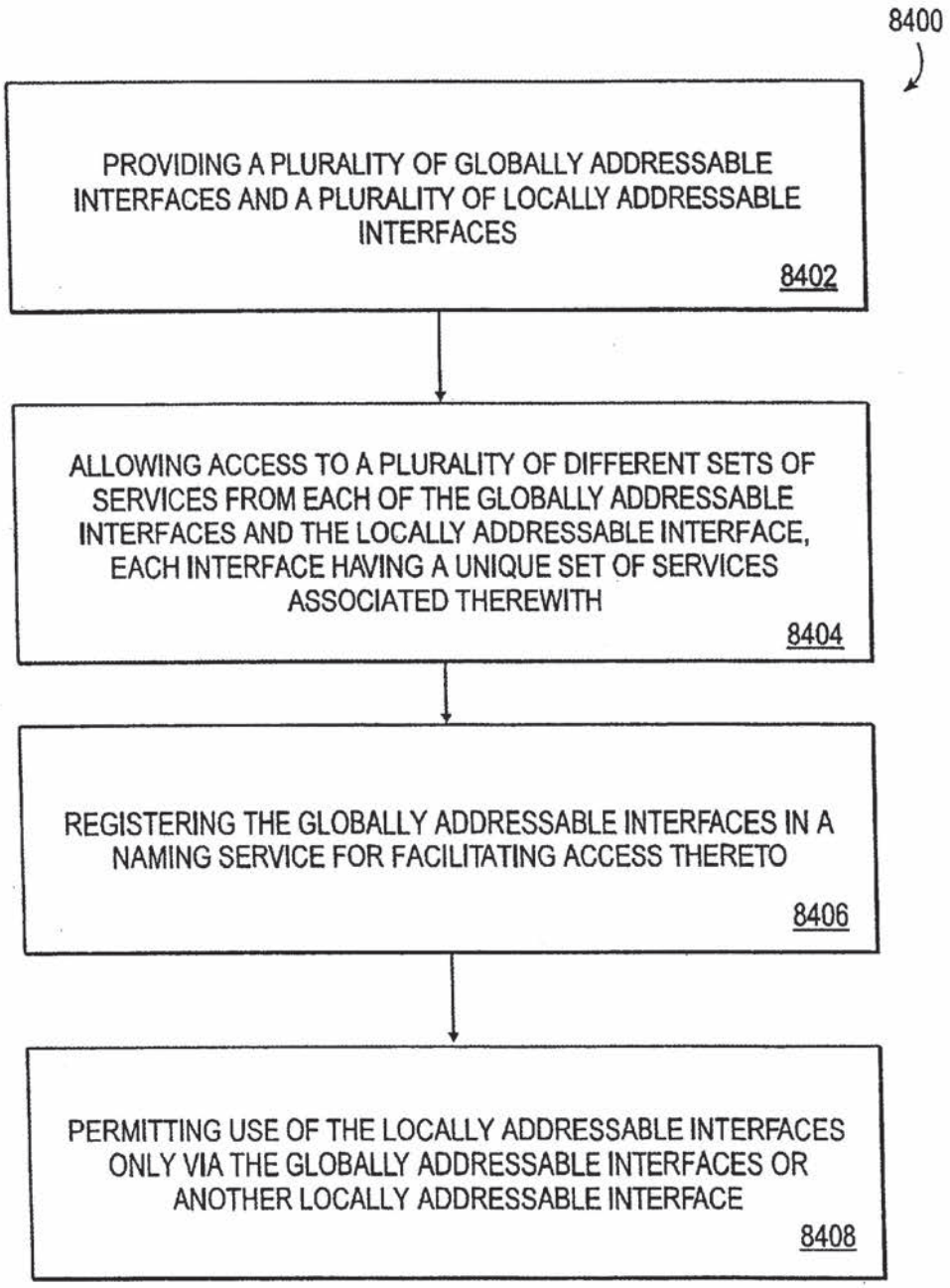

8400
PROVIDING A PLURALITY OF GLOBALLY ADDRESSABLE INTERFACES AND A PLURALITY OF LOCALLY ADDRESSABLE INTERFACES
8402
ALLOWING ACCESS TO A PLURALITY OF DIFFERENT SETS OF SERVICES FROM EACH OF THE GLOBALLY ADDRESSABLE INTERFACES AND THE LOCALLY ADDRESSABLE INTERFACE, EACH INTERFACE HAVING A UNIQUE SET OF SERVICES ASSOCIATED THEREWITH
8404
REGISTERING THE GLOBALLY ADDRESSABLE INTERFACES IN A NAMING SERVICE FOR FACILITATING ACCESS THERETO
8406
PERMITTING USE OF THE LOCALLY ADDRESSABLE INTERFACES ONLY VIA THE GLOBALLY ADDRESSABLE INTERFACES OR ANOTHER LOCALLY ADDRESSABLE INTERFACE
8408

Fig. 84

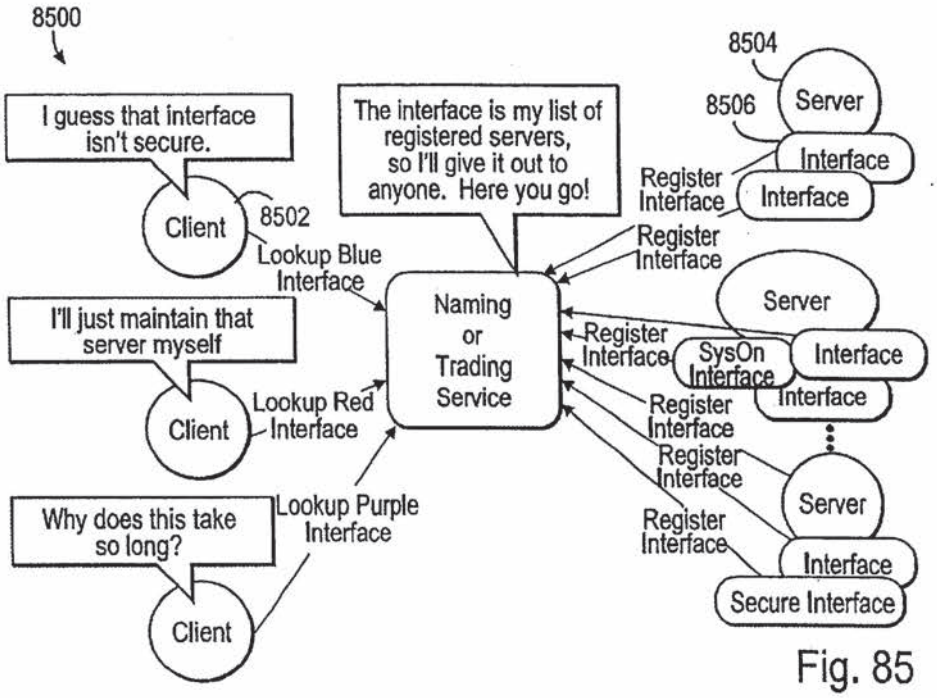

8500
I guess that interface isn't secure.
Client
8502
Lookup Blue Interface
The interface is my list of registered servers, so I'll give it out to anyone. Here you go!
Naming or Trading Service
I'll just maintain that server myself
Client
Lookup Red Interface
Why does this take so long?
Client
Lookup Purple Interface
8504
Server
8506
Interface
Register Interface
Interface
Register Interface
Server
Register Interface
SysOn Interface
Interface
Interface
Register Interface
Register Interface
Server
Register Interface
Interface
Secure Interface

Fig. 85

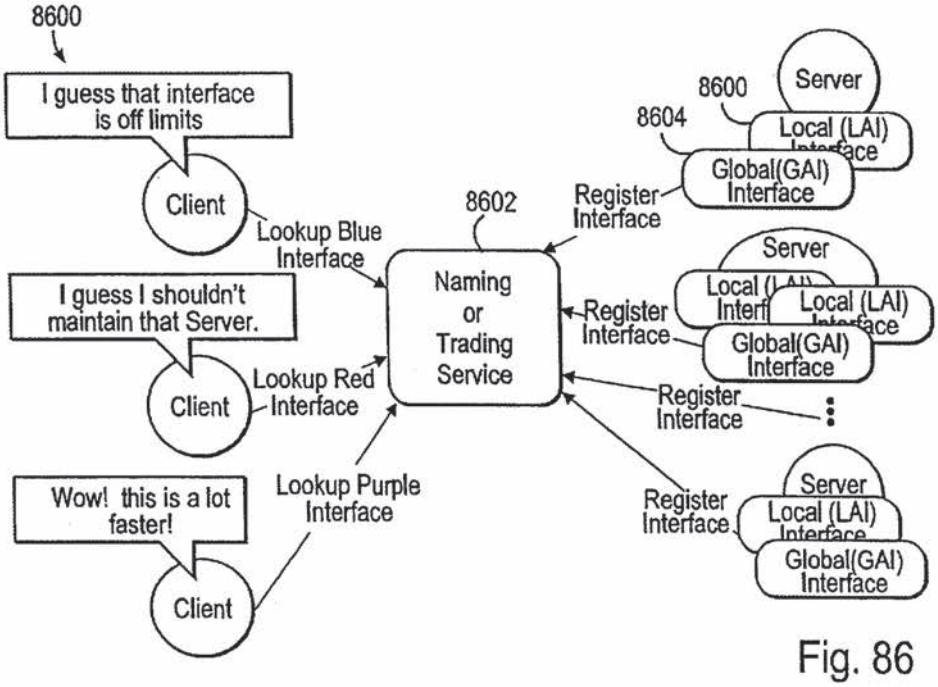

8600
I guess that interface is off limits
Client
Lookup Blue Interface
I guess I shouldn't maintain that Server.
Client
Lookup Red Interface
Wow! this is a lot faster!
Client
Lookup Purple Interface
8602
Naming or Trading Service
8600
Server
8604
Local (LAI) Interface
Global(GAI) Interface
Register Interface
Server
Local (LAI) Interface
Local (LAI) Interface
Global(GAI) Interface
Register Interface
Register Interface
Server
Register Interface
Local (LAI) Interface
Global(GAI) Interface

Fig. 86

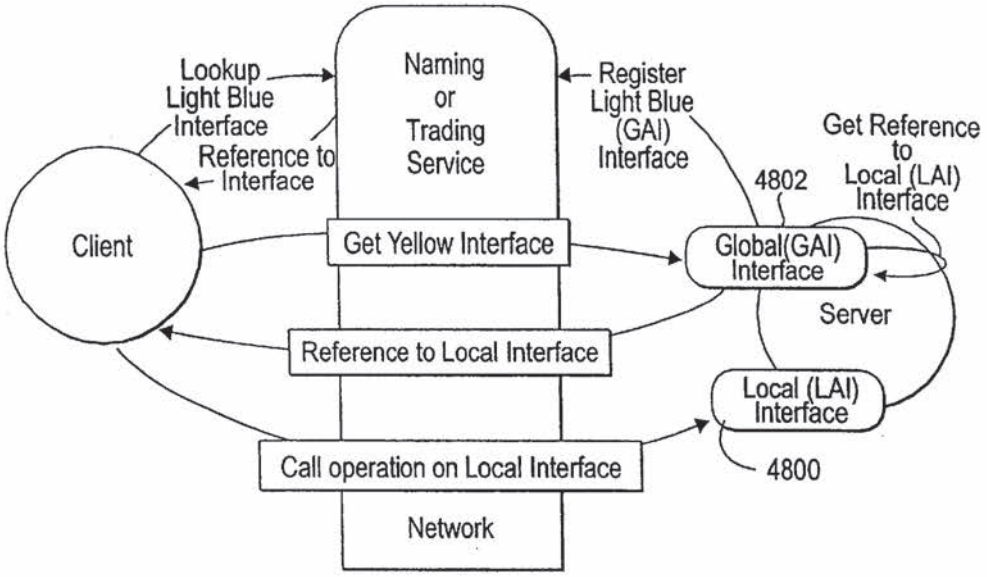

Lookup
Light Blue
Interface
Reference to
Interface
Naming
or
Trading
Service
Register
Light Blue
(GAI)
Interface
Get Reference
to
Local (LAI)
Interface
4802
Client
Get Yellow Interface
Global(GAI)
Interface
Server
Reference to Local Interface
Local (LAI)
Interface
Call operation on Local Interface
4800
Network

Fig. 87

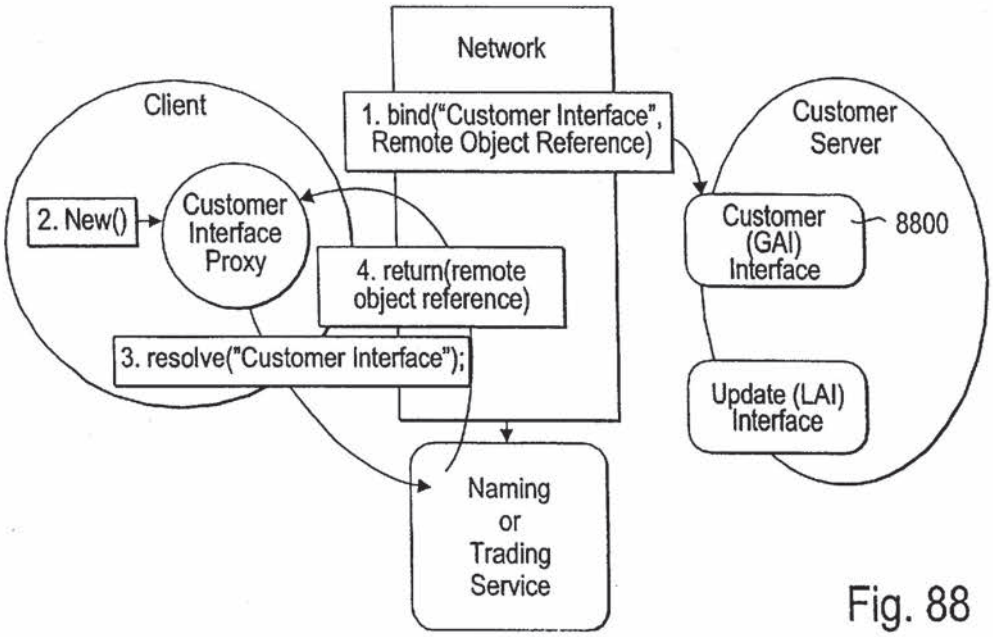

Network
Client
1. bind("Customer Interface",
Remote Object Reference)
Customer
Server
2. New()
Customer
Interface
Proxy
Customer
(GAI)
Interface
8800
4. return(remote
object reference)
3. resolve("Customer Interface");
Update (LAI)
Interface
Naming
or
Trading
Service

Fig. 88

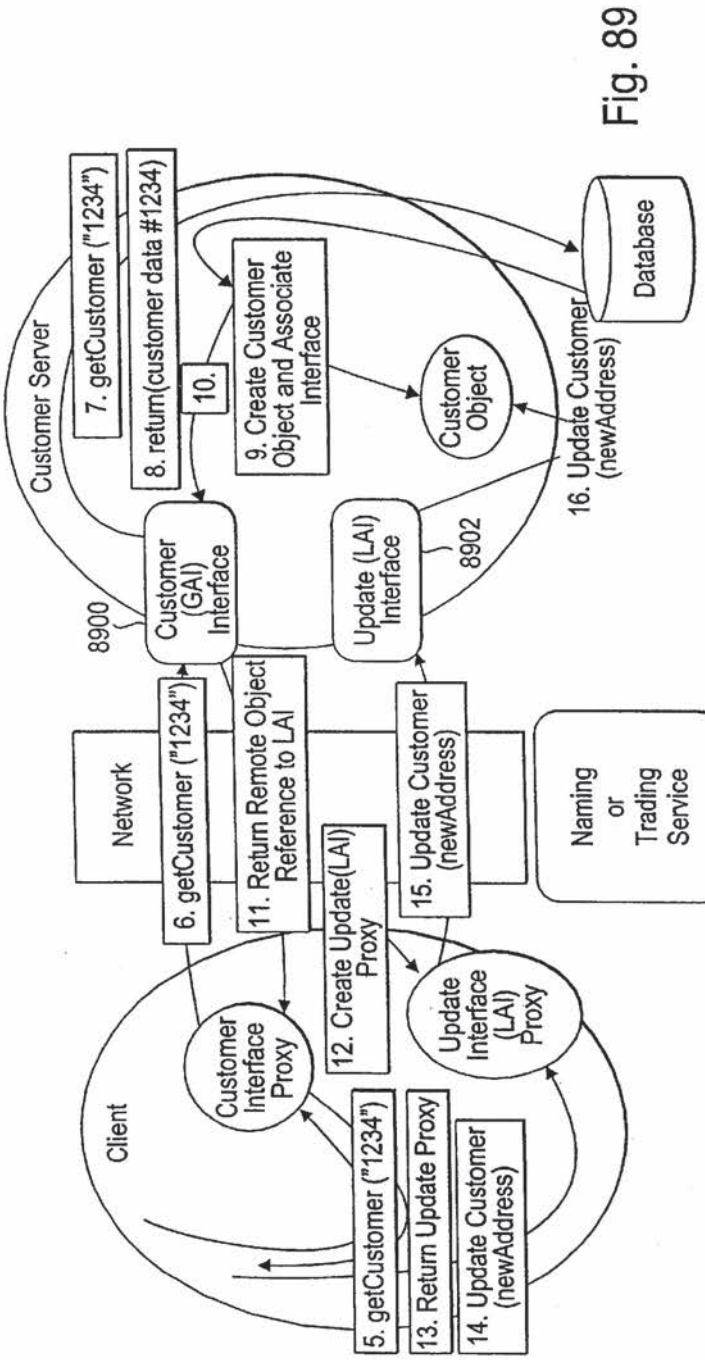

Customer Server
7. getCustomer ("1234")
8. return(customer data #1234)
10.
9. Create Customer Object and Associate Interface
Customer Object
Database
16. Update Customer (newAddress)
8900
Customer (GAI) Interface
Update (LAI) Interface
8902
Network
6. getCustomer ("1234")
11. Return Remote Object Reference to LAI
12. Create Update(LAI) Proxy
15. Update Customer (newAddress)
Naming or Trading Service
Client
Customer Interface Proxy
Update Interface (LAI) Proxy
5. getCustomer ("1234")
13. Return Update Proxy
14. Update Customer (newAddress)

Fig. 89

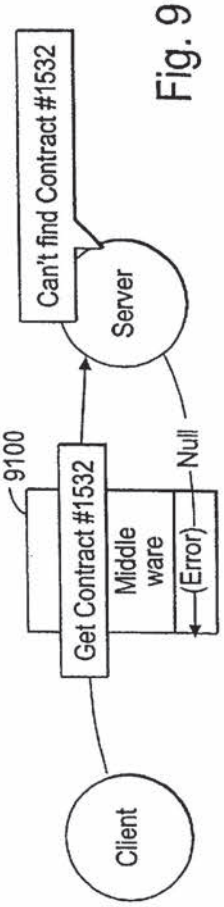

Can't find Contract #1532
Server
9100
Get Contract #1532
Middle ware
Null
(Error)
Client

Fig. 91

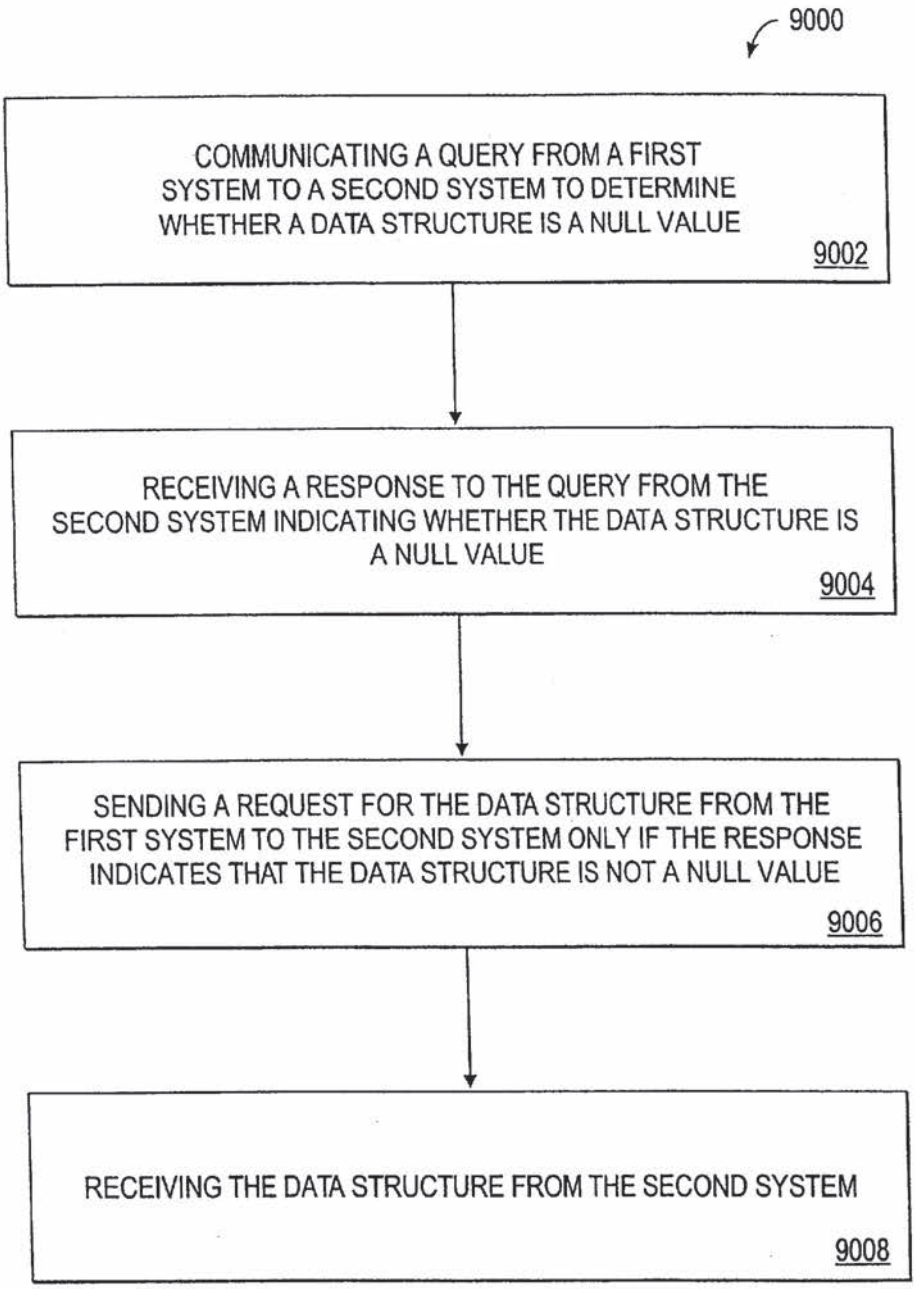

9000
COMMUNICATING A QUERY FROM A FIRST SYSTEM TO A SECOND SYSTEM TO DETERMINE WHETHER A DATA STRUCTURE IS A NULL VALUE
9002
RECEIVING A RESPONSE TO THE QUERY FROM THE SECOND SYSTEM INDICATING WHETHER THE DATA STRUCTURE IS A NULL VALUE
9004
SENDING A REQUEST FOR THE DATA STRUCTURE FROM THE FIRST SYSTEM TO THE SECOND SYSTEM ONLY IF THE RESPONSE INDICATES THAT THE DATA STRUCTURE IS NOT A NULL VALUE
9006
RECEIVING THE DATA STRUCTURE FROM THE SECOND SYSTEM
9008

Fig. 90

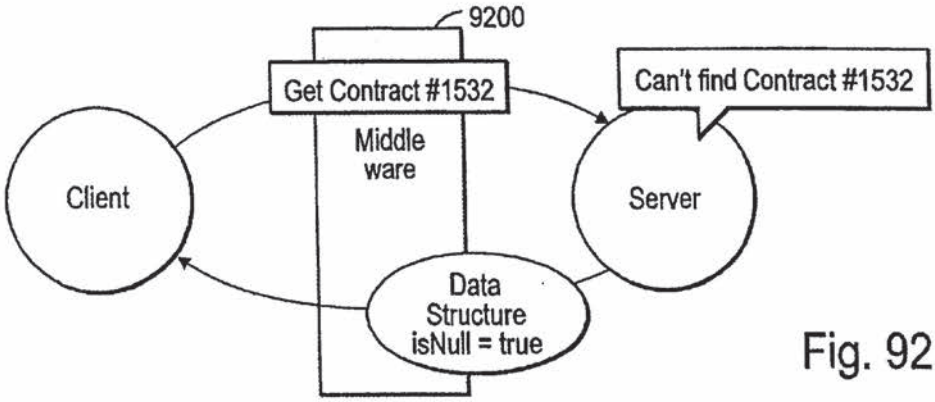


Fig. 92

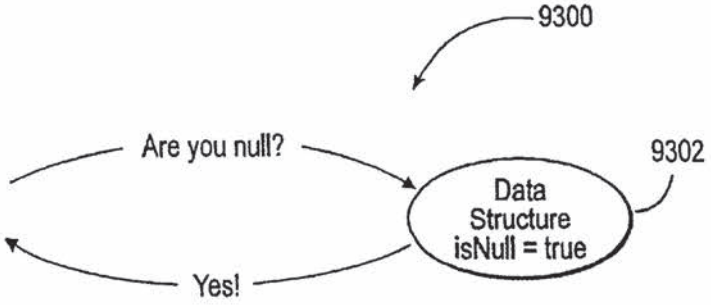


Fig. 93

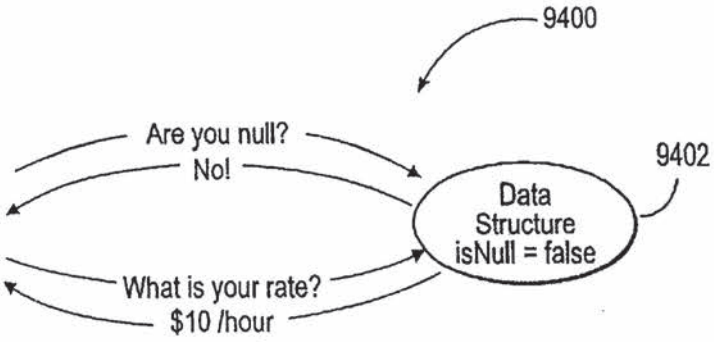


Fig. 94

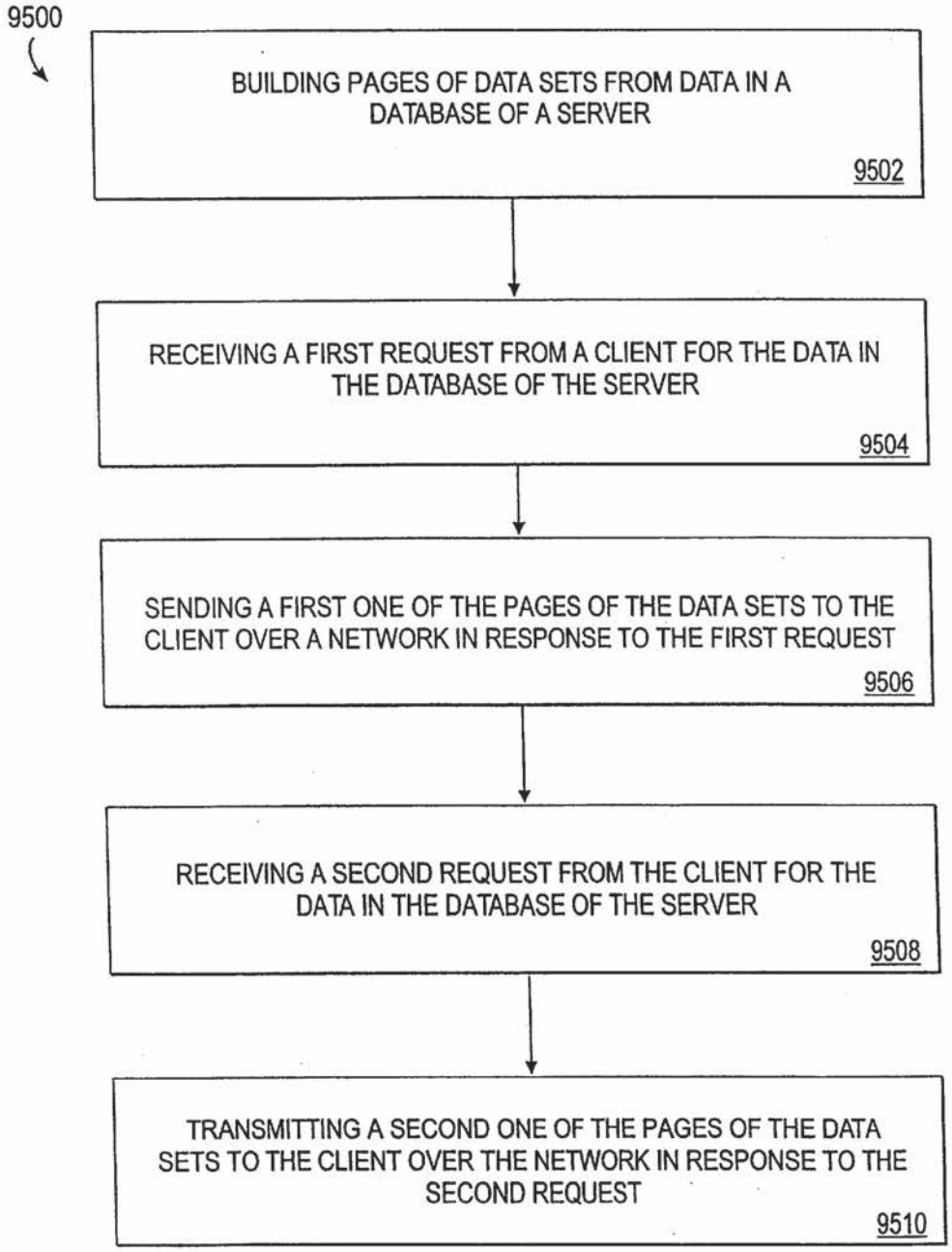

9500
BUILDING PAGES OF DATA SETS FROM DATA IN A DATABASE OF A SERVER
9502
RECEIVING A FIRST REQUEST FROM A CLIENT FOR THE DATA IN THE DATABASE OF THE SERVER
9504
SENDING A FIRST ONE OF THE PAGES OF THE DATA SETS TO THE CLIENT OVER A NETWORK IN RESPONSE TO THE FIRST REQUEST
9506
RECEIVING A SECOND REQUEST FROM THE CLIENT FOR THE DATA IN THE DATABASE OF THE SERVER
9508
TRANSMITTING A SECOND ONE OF THE PAGES OF THE DATA SETS TO THE CLIENT OVER THE NETWORK IN RESPONSE TO THE SECOND REQUEST
9510

Fig. 95

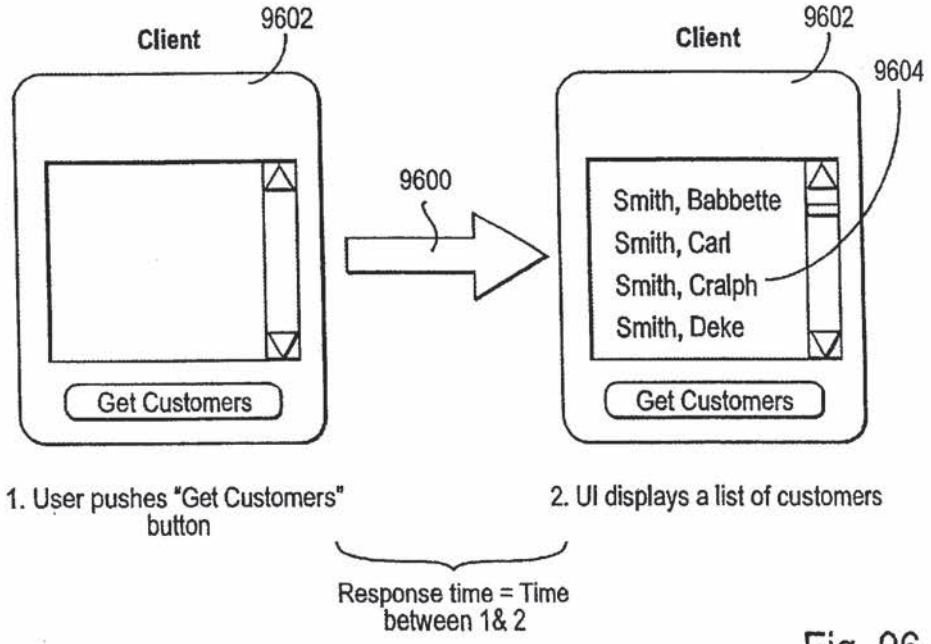

Client
9602
Get Customers
9600
Client
9602
9604
Smith, Babbette
Smith, Carl
Smith, Cralph
Smith, Deke
Get Customers
1. User pushes "Get Customers" button
2. UI displays a list of customers
Response time = Time between 1& 2

Fig. 96

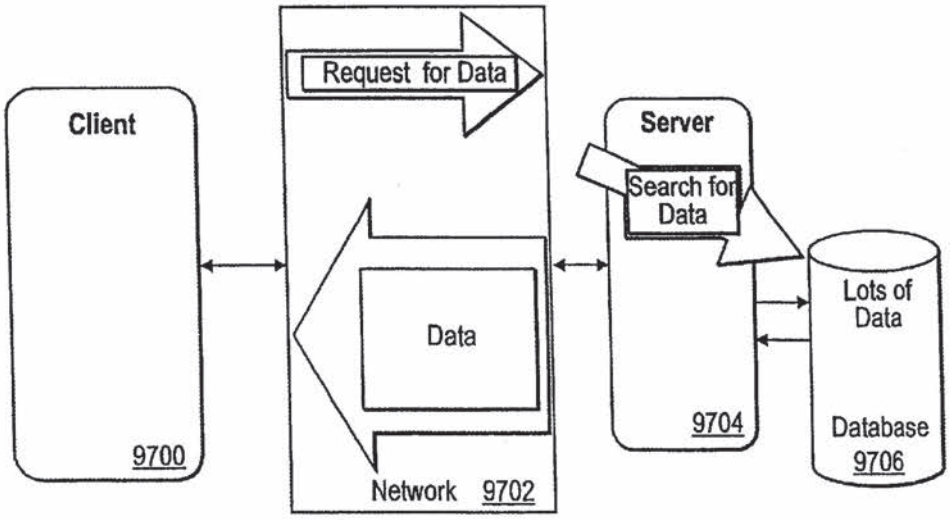

Client
9700
Request for Data
Data
Network 9702
Server
Search for Data
9704
Lots of Data
Database
9706

Fig. 97

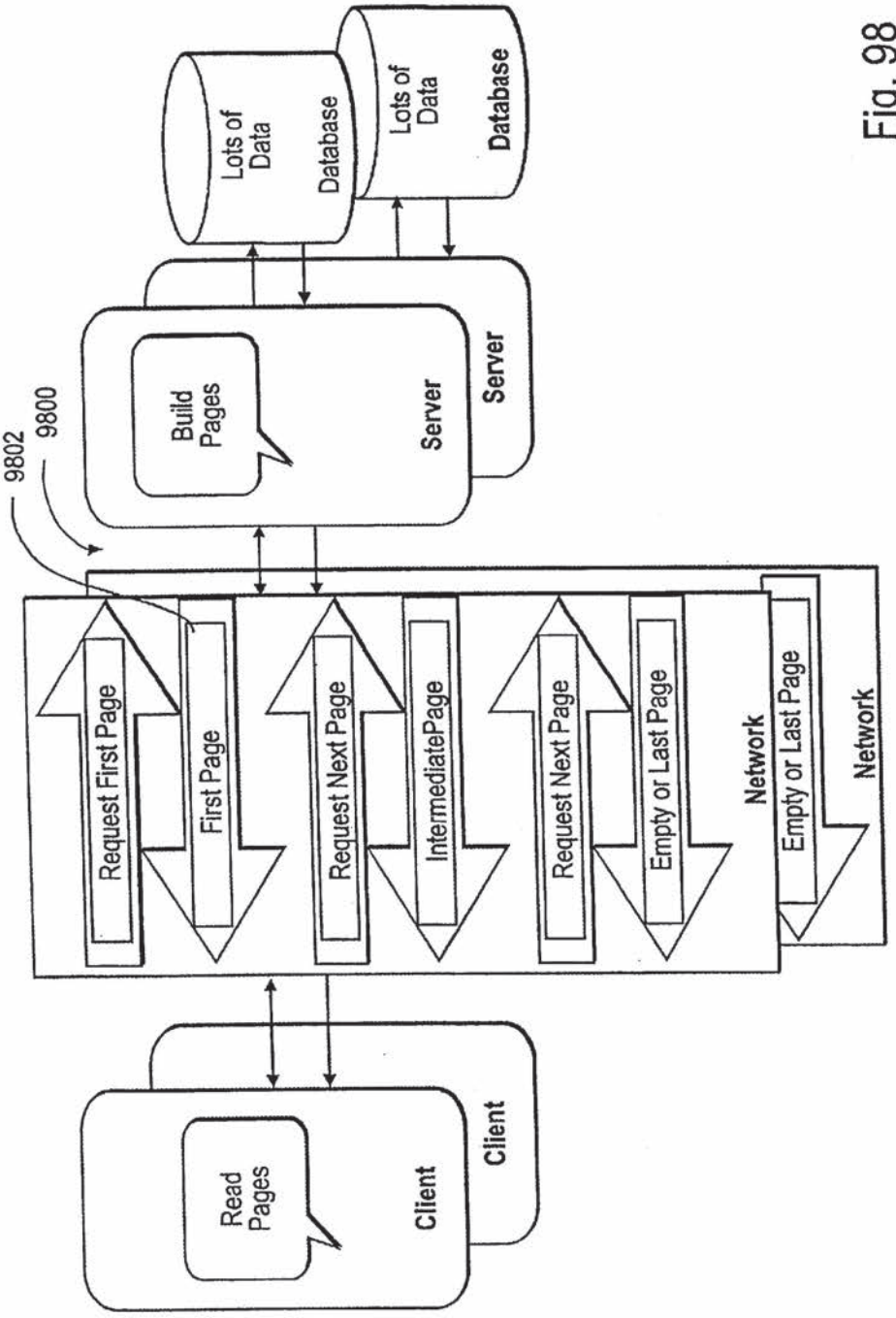

Lots of Data
Database
Lots of Data
Database
Build Pages
Server
Server
9802
9800
Request First Page
First Page
Request Next Page
IntermediatePage
Request Next Page
Empty or Last Page
Network
Empty or Last Page
Network
Read Pages
Client
Client

Fig. 98

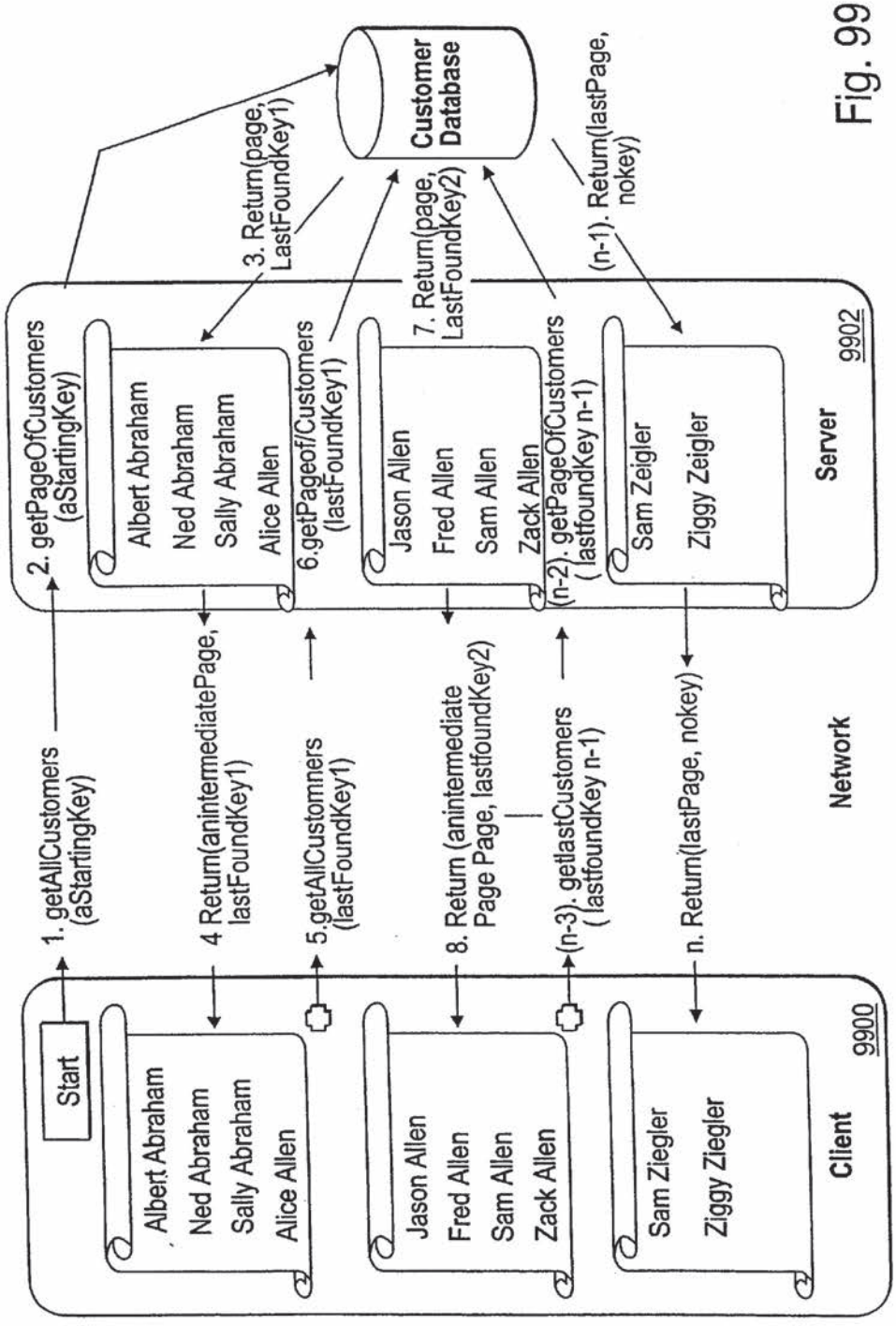

Customer Database
3. Return(page, LastFoundKey1)
7. Return(page, LastFoundKey2)
(n-1). Return(lastPage, nokey)
2. getPageOfCustomers (aStartingKey)
Albert Abraham
Ned Abraham
Sally Abraham
Alice Allen
6.getPageof/Customers (lastFoundKey1)
Jason Allen
Fred Allen
Sam Allen
Zack Allen
(n-2). getPageOfCustomers (lastfoundKey n-1)
Sam Zeigler
Ziggy Zeigler
Server 9902
1. getAllCustomers (aStartingKey)
4 Return(anintermediatePage, lastFoundKey1)
5.getAllCustomners (lastFoundKey1)
8. Return (anintermediate Page Page, lastfoundKey2)
(n-3). getlastCustomers (lastfoundKey n-1)
n. Return(lastPage, nokey)
Network
Start
Albert Abraham
Ned Abraham
Sally Abraham
Alice Allen
Jason Allen
Fred Allen
Sam Allen
Zack Allen
Sam Ziegler
Ziggy Ziegler
Client 9900

Fig. 99

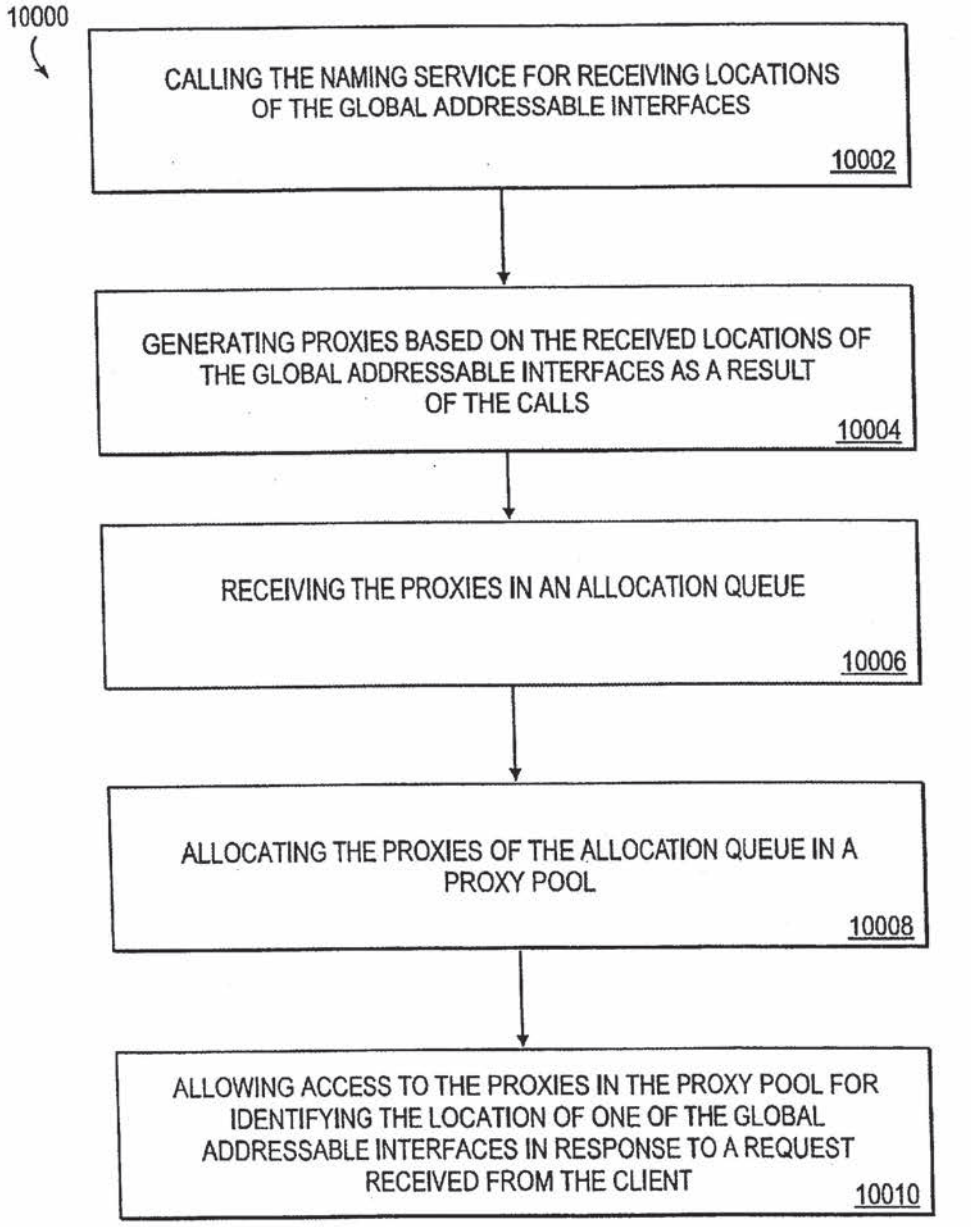

10000
CALLING THE NAMING SERVICE FOR RECEIVING LOCATIONS OF THE GLOBAL ADDRESSABLE INTERFACES
10002
GENERATING PROXIES BASED ON THE RECEIVED LOCATIONS OF THE GLOBAL ADDRESSABLE INTERFACES AS A RESULT OF THE CALLS
10004
RECEIVING THE PROXIES IN AN ALLOCATION QUEUE
10006
ALLOCATING THE PROXIES OF THE ALLOCATION QUEUE IN A PROXY POOL
10008
ALLOWING ACCESS TO THE PROXIES IN THE PROXY POOL FOR IDENTIFYING THE LOCATION OF ONE OF THE GLOBAL ADDRESSABLE INTERFACES IN RESPONSE TO A REQUEST RECEIVED FROM THE CLIENT
10010

Fig. 100

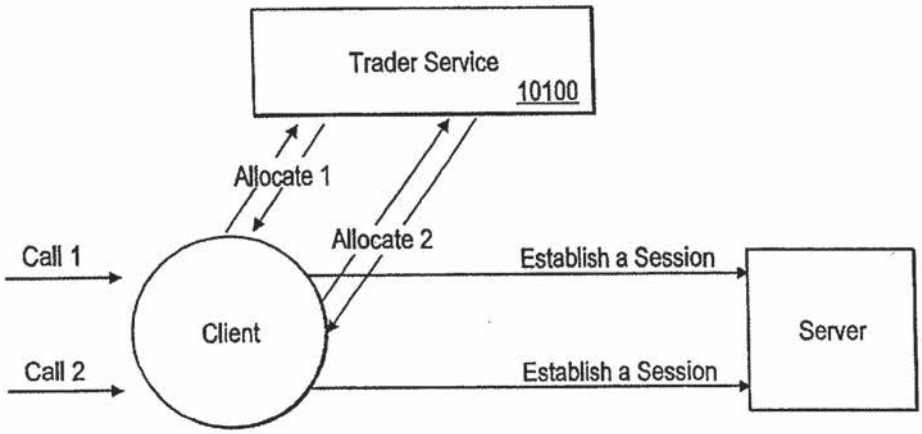


Fig. 101

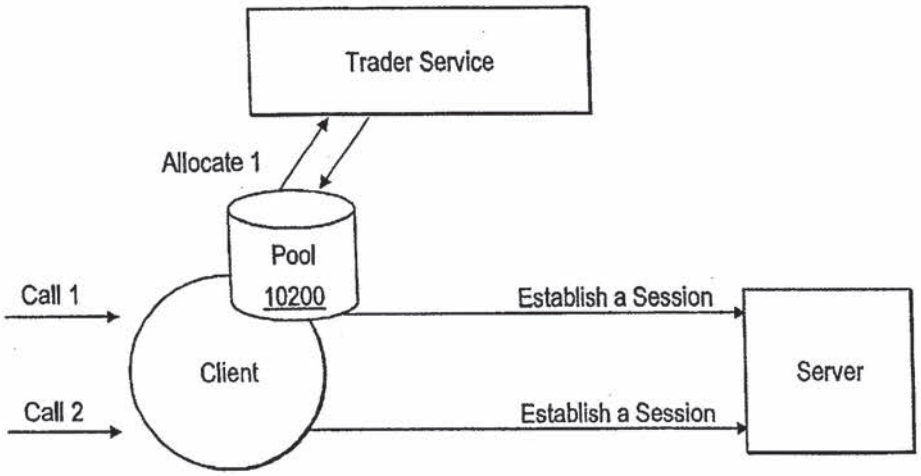


Fig. 102

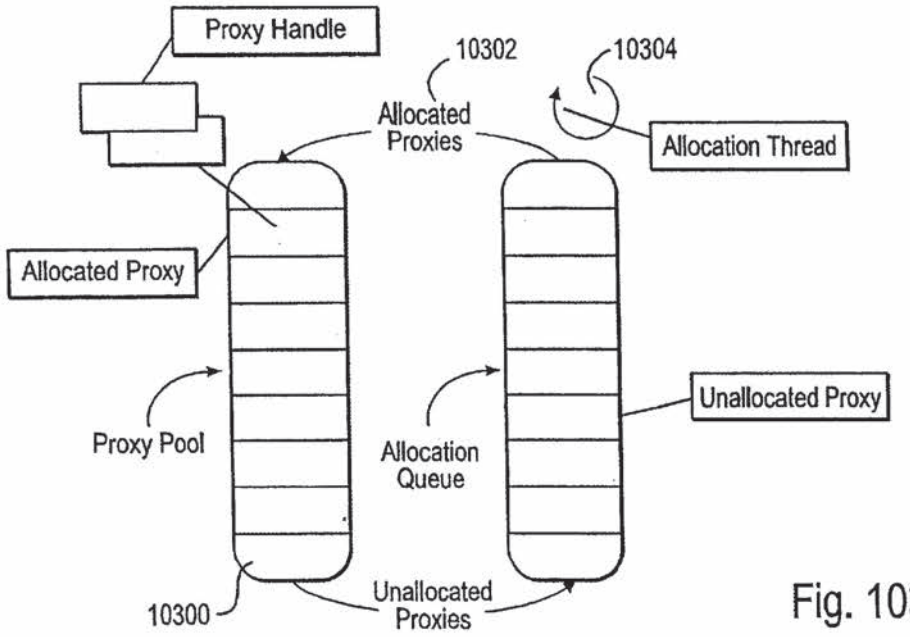


Fig. 103

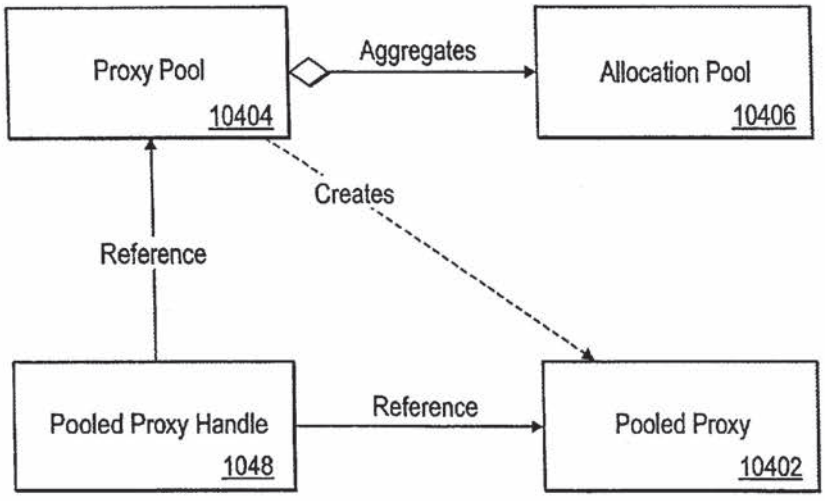


Fig. 104

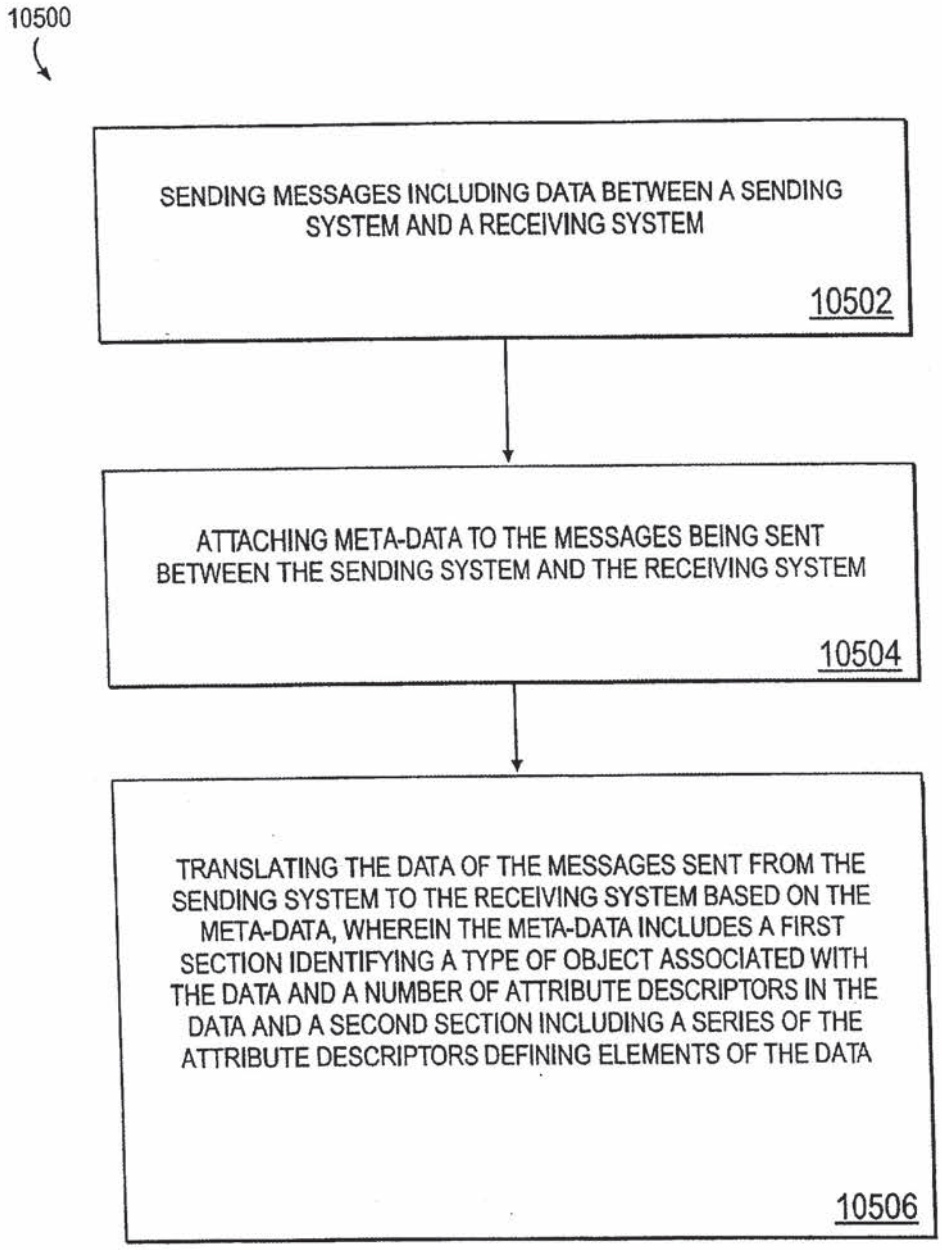

10500
SENDING MESSAGES INCLUDING DATA BETWEEN A SENDING SYSTEM AND A RECEIVING SYSTEM
10502
ATTACHING META-DATA TO THE MESSAGES BEING SENT BETWEEN THE SENDING SYSTEM AND THE RECEIVING SYSTEM
10504
TRANSLATING THE DATA OF THE MESSAGES SENT FROM THE SENDING SYSTEM TO THE RECEIVING SYSTEM BASED ON THE META-DATA, WHEREIN THE META-DATA INCLUDES A FIRST SECTION IDENTIFYING A TYPE OF OBJECT ASSOCIATED WITH THE DATA AND A NUMBER OF ATTRIBUTE DESCRIPTORS IN THE DATA AND A SECOND SECTION INCLUDING A SERIES OF THE ATTRIBUTE DESCRIPTORS DEFINING ELEMENTS OF THE DATA
10506

Fig. 105

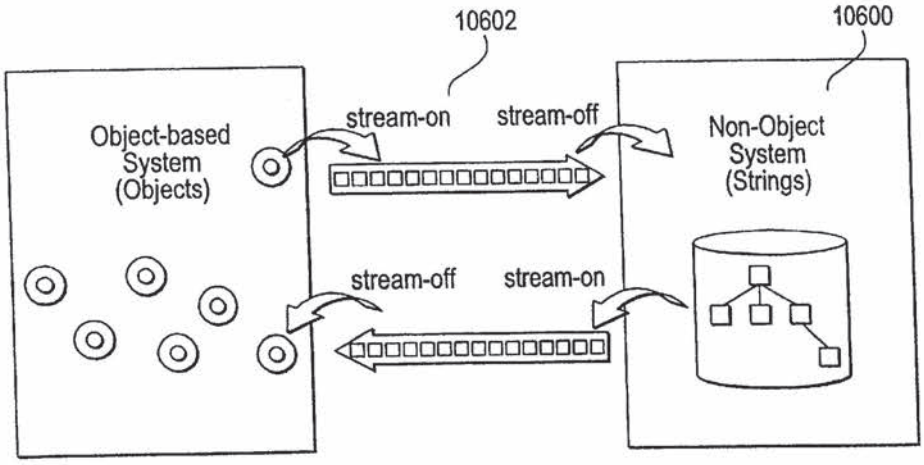


Fig. 106

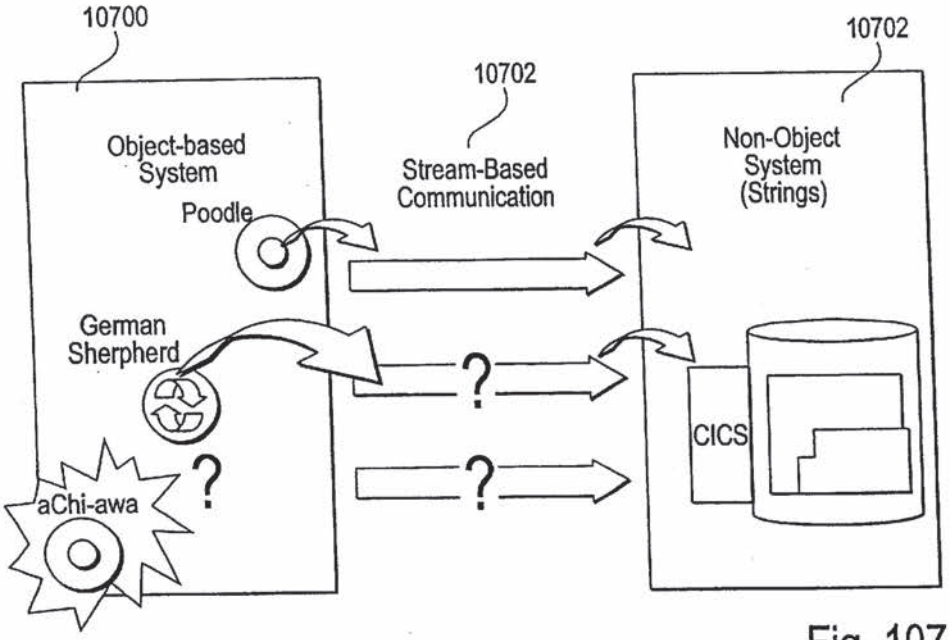


Fig. 107

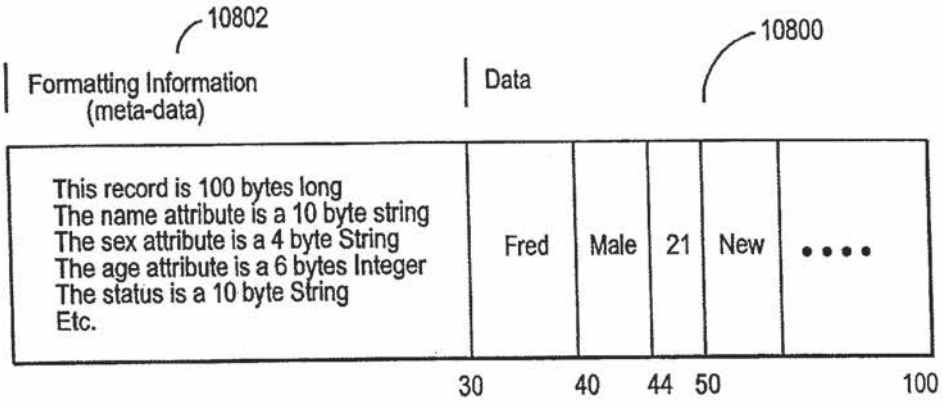


Fig. 108

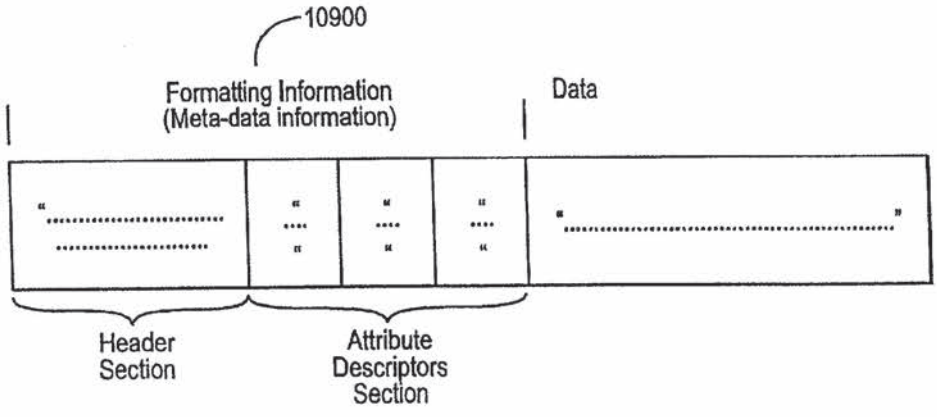


Fig. 109

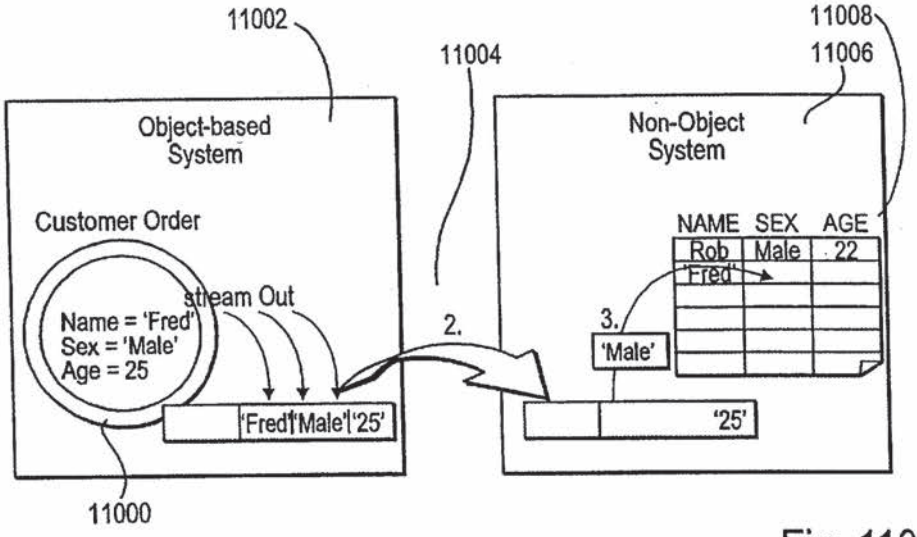


Fig. 110

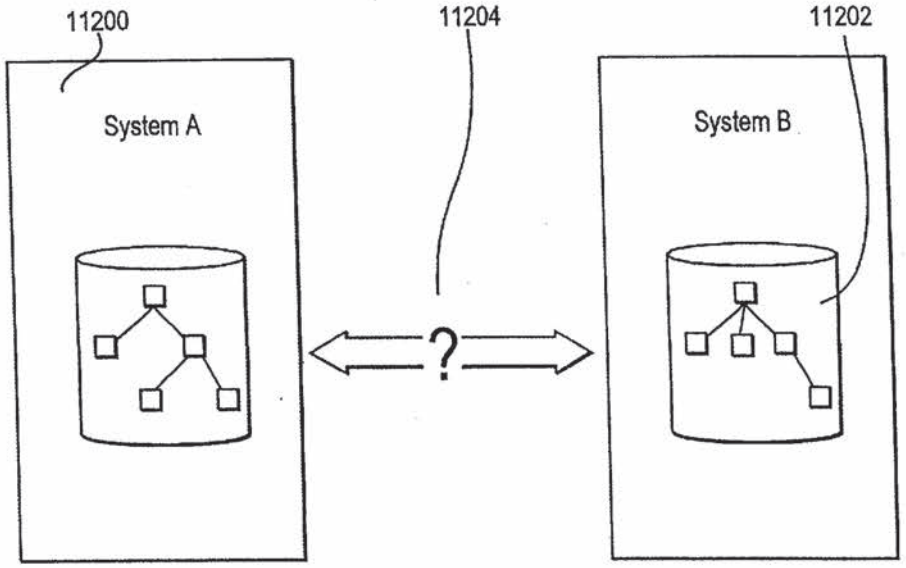


Fig. 112

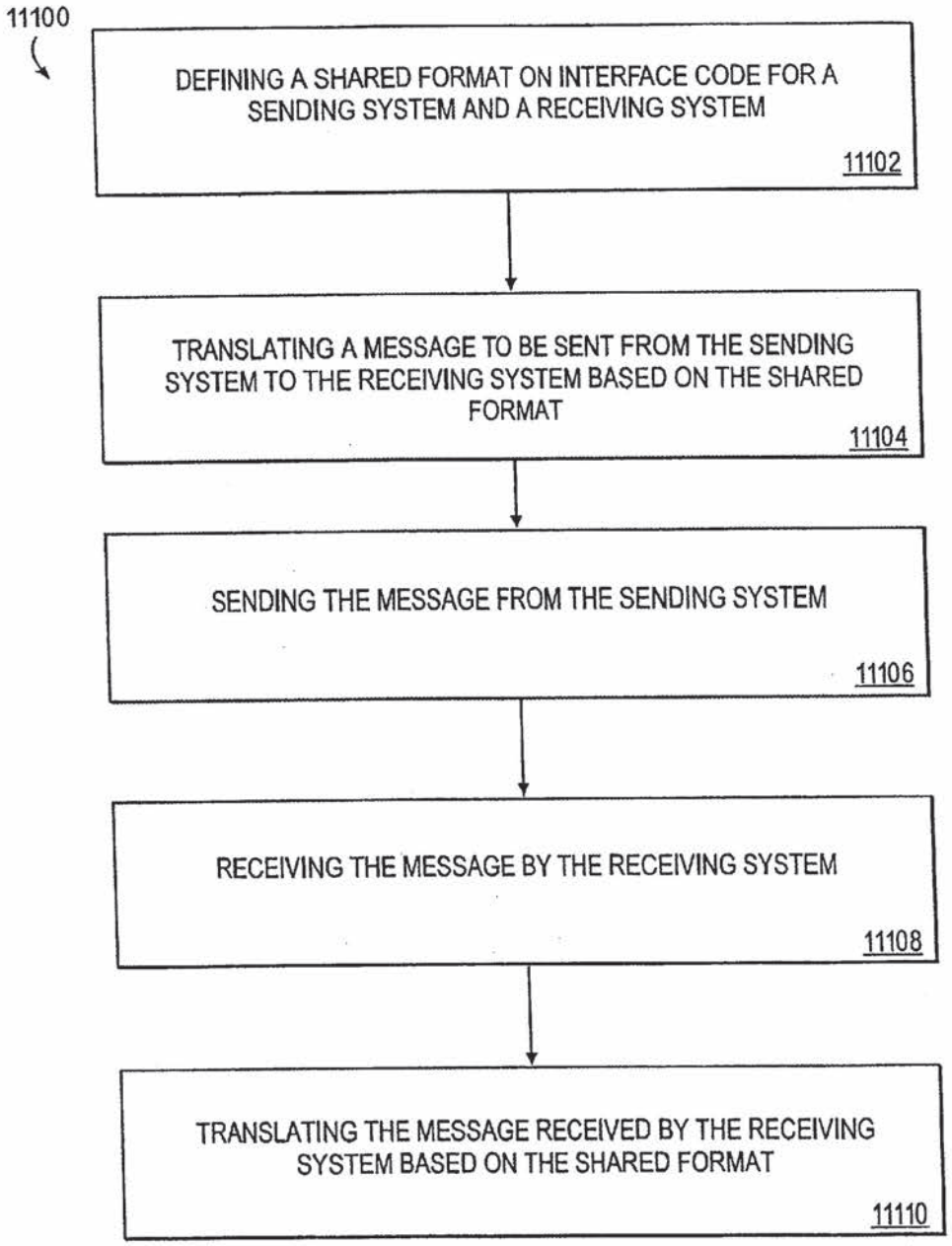

11100
DEFINING A SHARED FORMAT ON INTERFACE CODE FOR A SENDING SYSTEM AND A RECEIVING SYSTEM
11102
TRANSLATING A MESSAGE TO BE SENT FROM THE SENDING SYSTEM TO THE RECEIVING SYSTEM BASED ON THE SHARED FORMAT
11104
SENDING THE MESSAGE FROM THE SENDING SYSTEM
11106
RECEIVING THE MESSAGE BY THE RECEIVING SYSTEM
11108
TRANSLATING THE MESSAGE RECEIVED BY THE RECEIVING SYSTEM BASED ON THE SHARED FORMAT
11110

Fig. 111

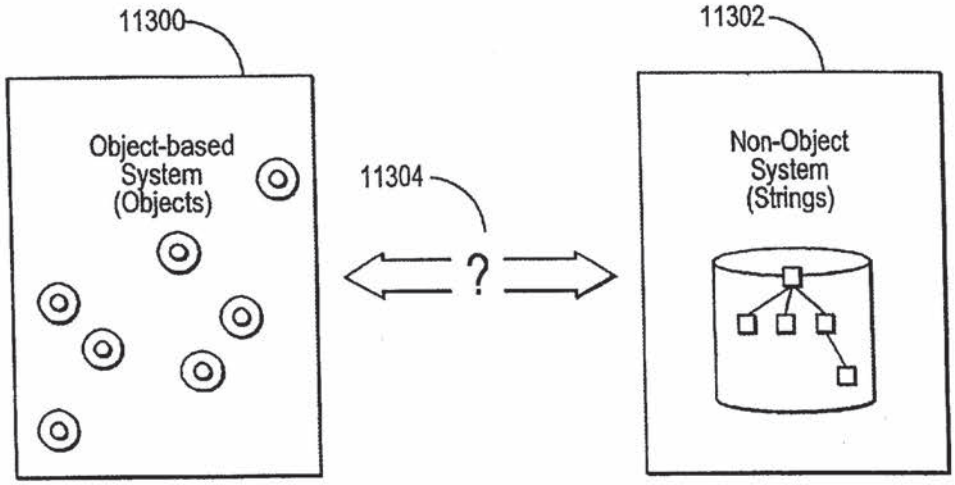


Fig. 113

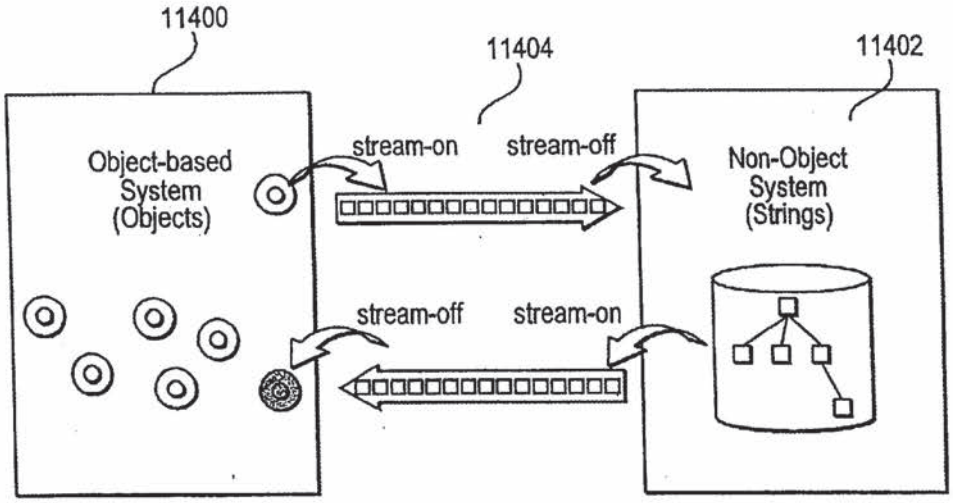


Fig. 114

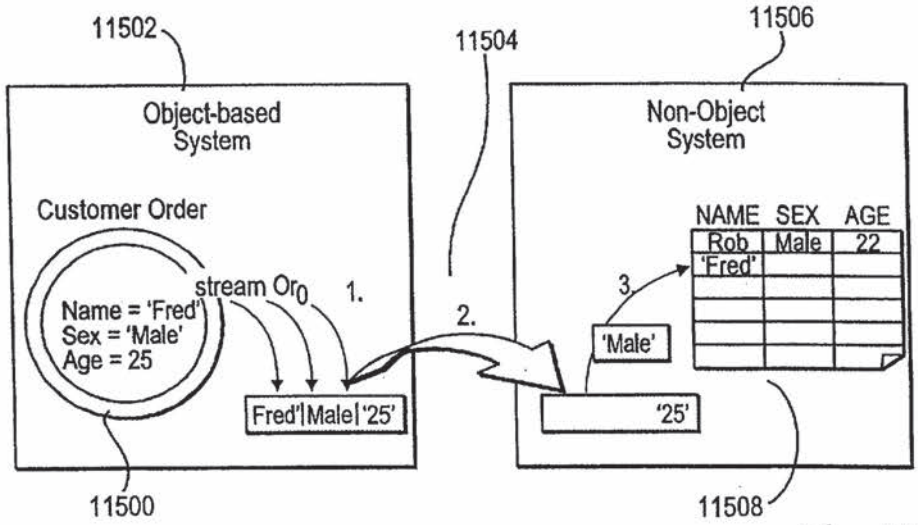


Fig. 115

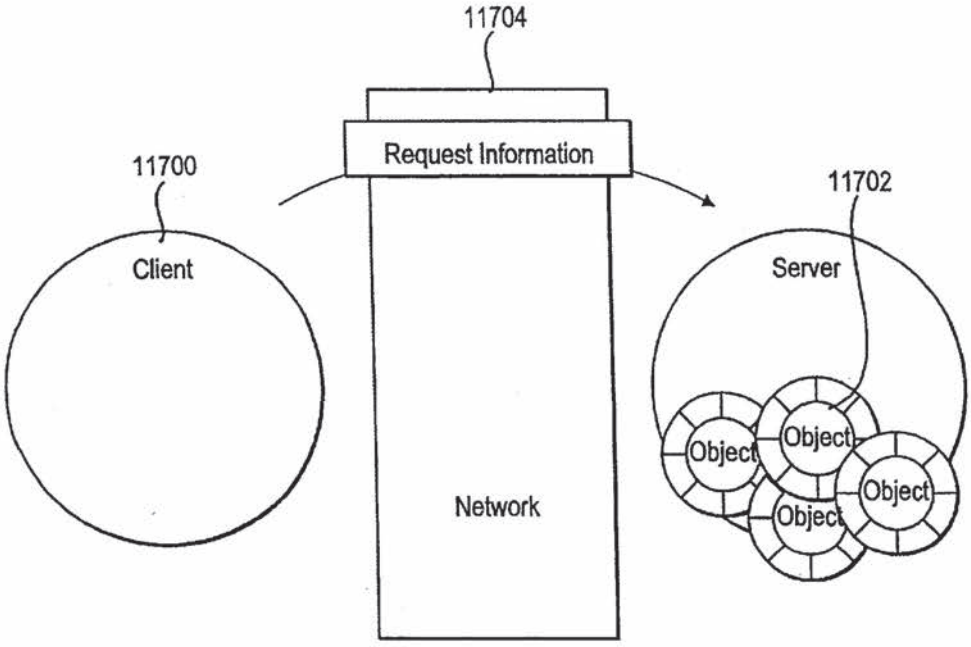


Fig. 117

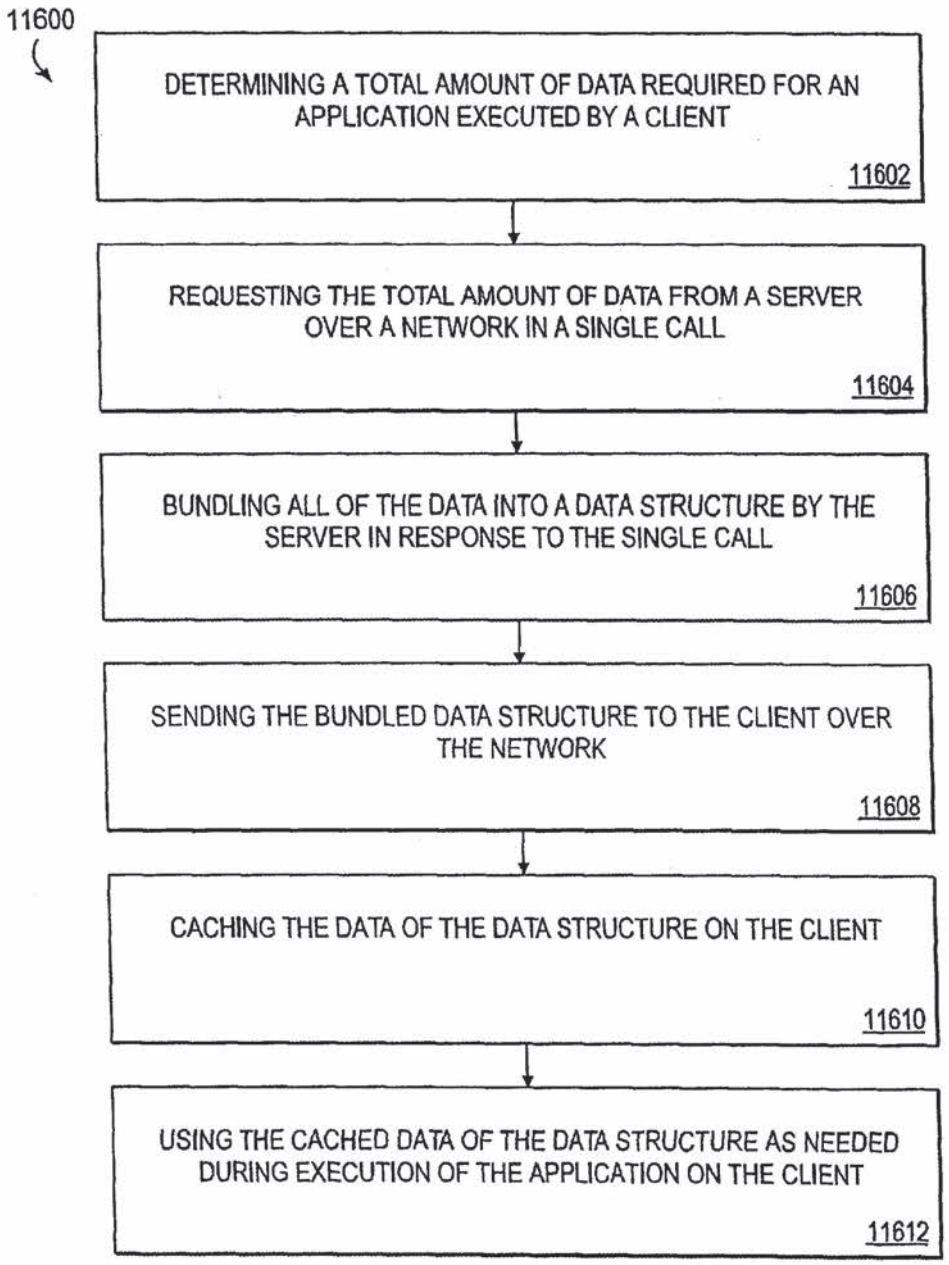

11600
DETERMINING A TOTAL AMOUNT OF DATA REQUIRED FOR AN APPLICATION EXECUTED BY A CLIENT
11602
REQUESTING THE TOTAL AMOUNT OF DATA FROM A SERVER OVER A NETWORK IN A SINGLE CALL
11604
BUNDLING ALL OF THE DATA INTO A DATA STRUCTURE BY THE SERVER IN RESPONSE TO THE SINGLE CALL
11606
SENDING THE BUNDLED DATA STRUCTURE TO THE CLIENT OVER THE NETWORK
11608
CACHING THE DATA OF THE DATA STRUCTURE ON THE CLIENT
11610
USING THE CACHED DATA OF THE DATA STRUCTURE AS NEEDED DURING EXECUTION OF THE APPLICATION ON THE CLIENT
11612

Fig. 116

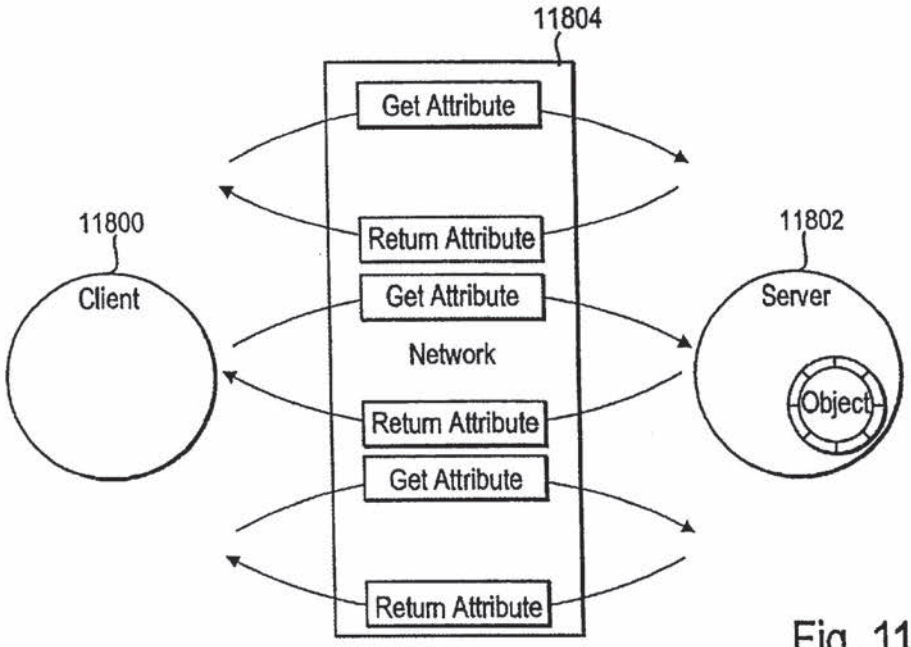


Fig. 118

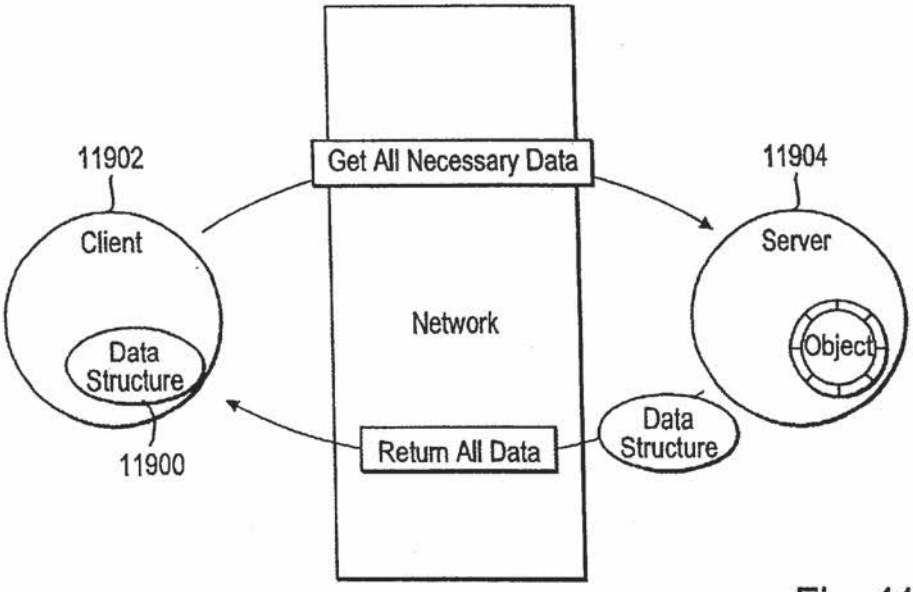


Fig. 119

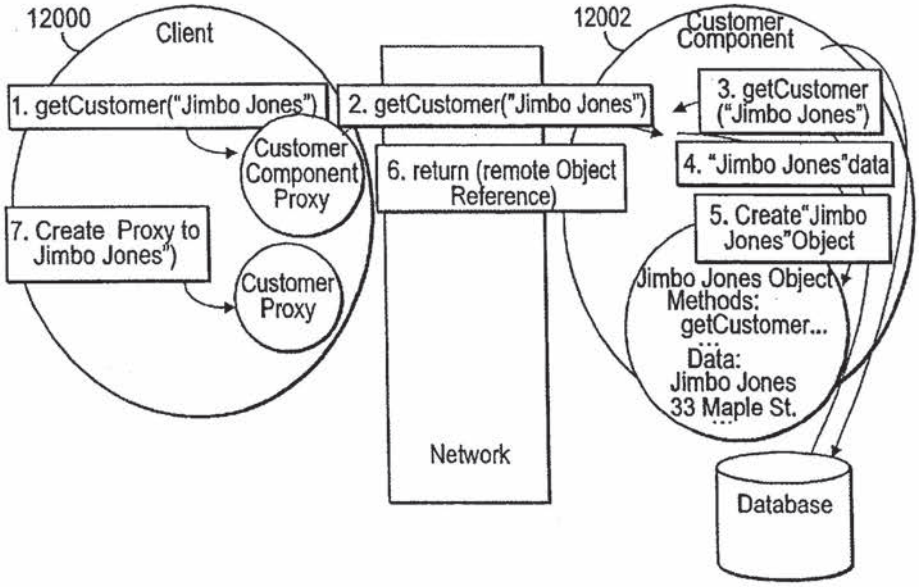


Fig. 120

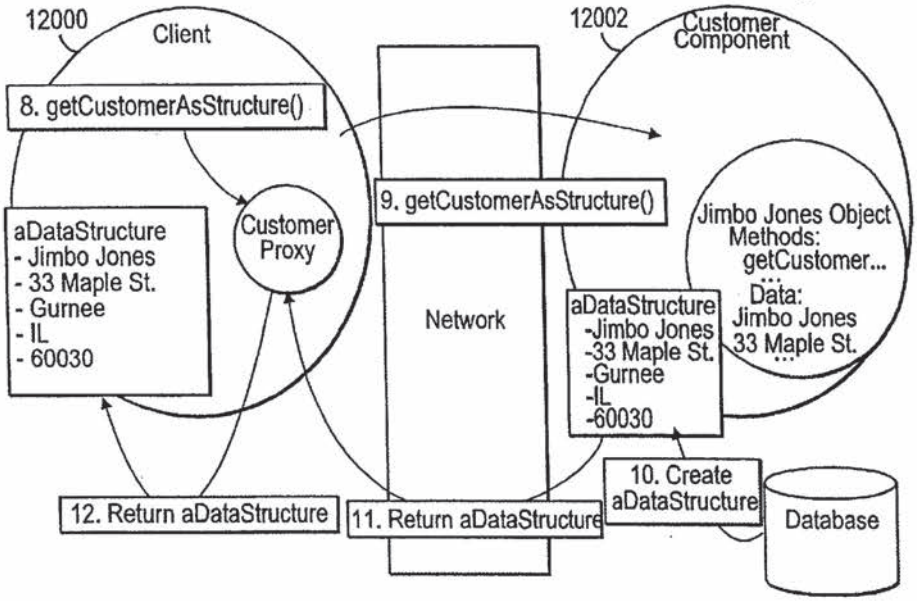


Fig. 121

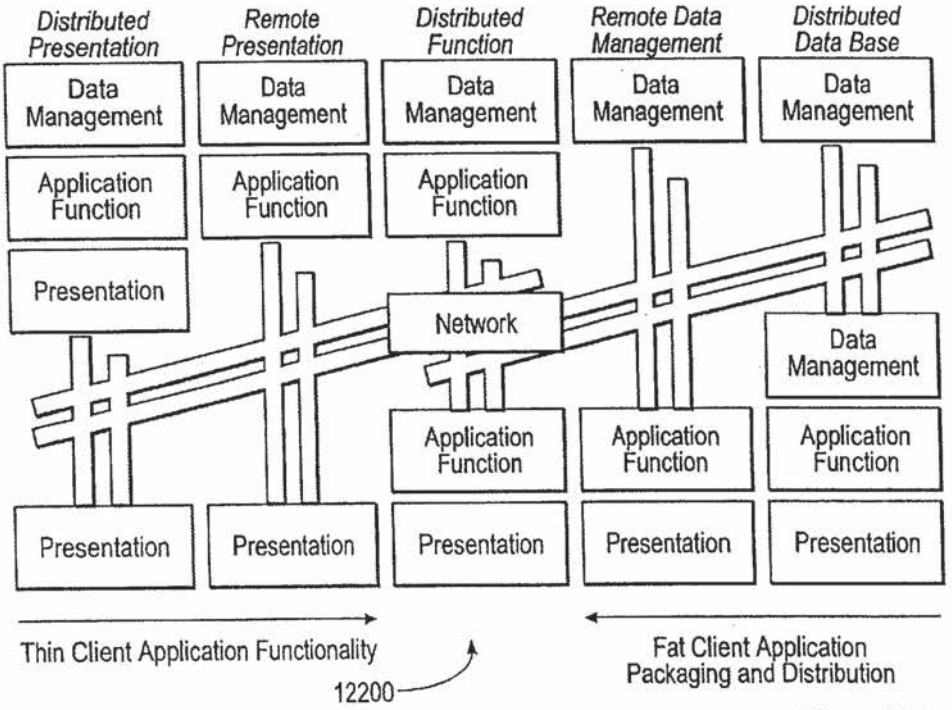


Fig. 122

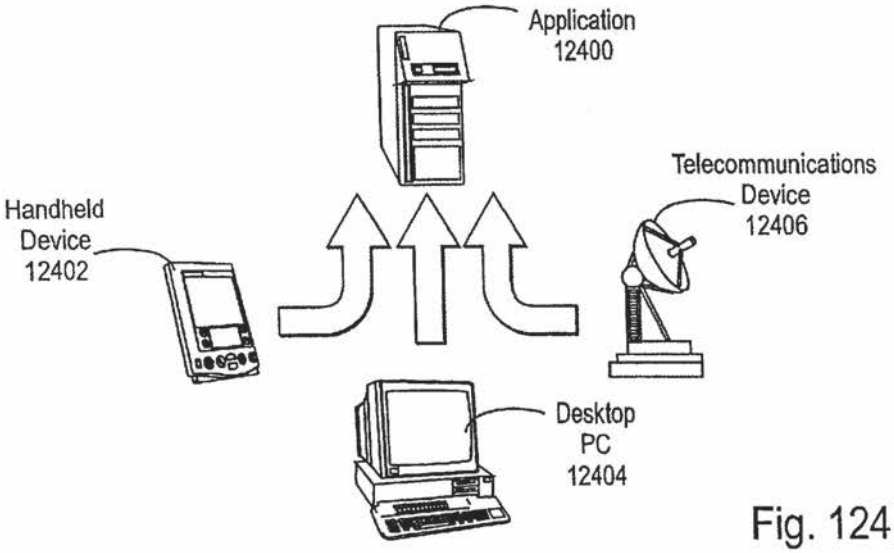


Fig. 124

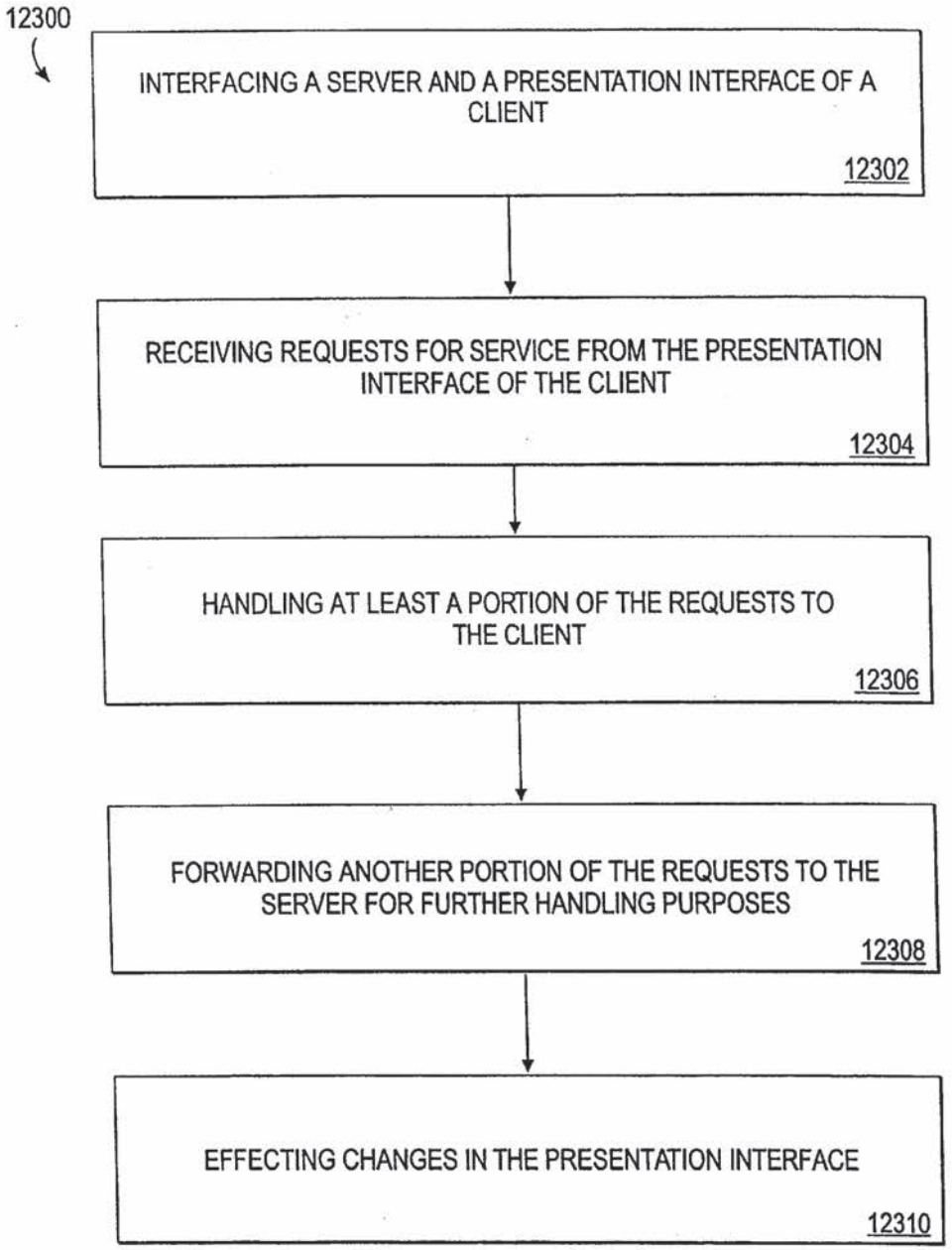

12300
INTERFACING A SERVER AND A PRESENTATION INTERFACE OF A CLIENT
12302
RECEIVING REQUESTS FOR SERVICE FROM THE PRESENTATION INTERFACE OF THE CLIENT
12304
HANDLING AT LEAST A PORTION OF THE REQUESTS TO THE CLIENT
12306
FORWARDING ANOTHER PORTION OF THE REQUESTS TO THE SERVER FOR FURTHER HANDLING PURPOSES
12308
EFFECTING CHANGES IN THE PRESENTATION INTERFACE
12310

Fig. 123

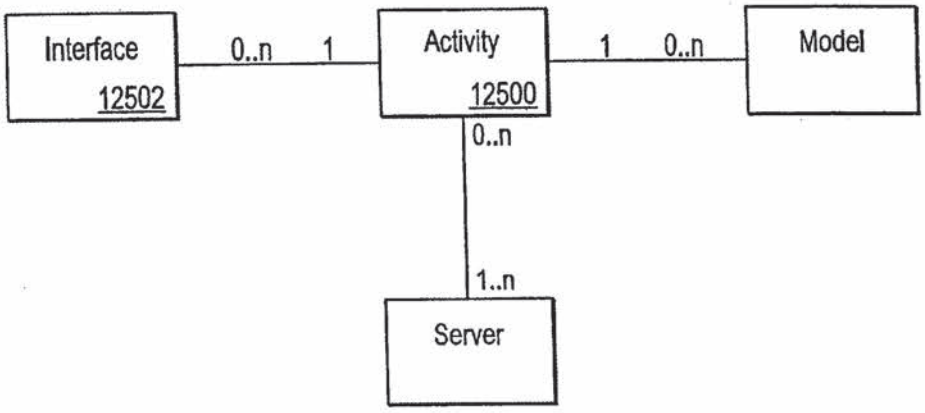


Fig. 125

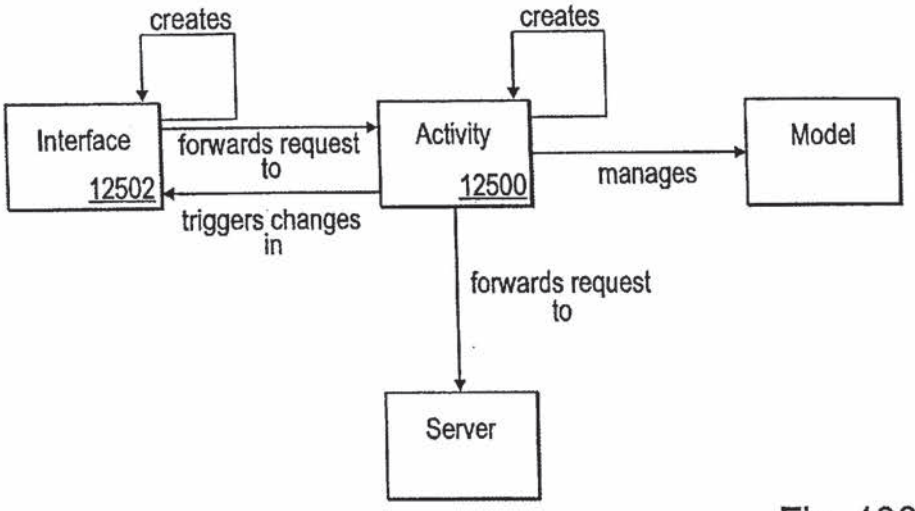


Fig. 126

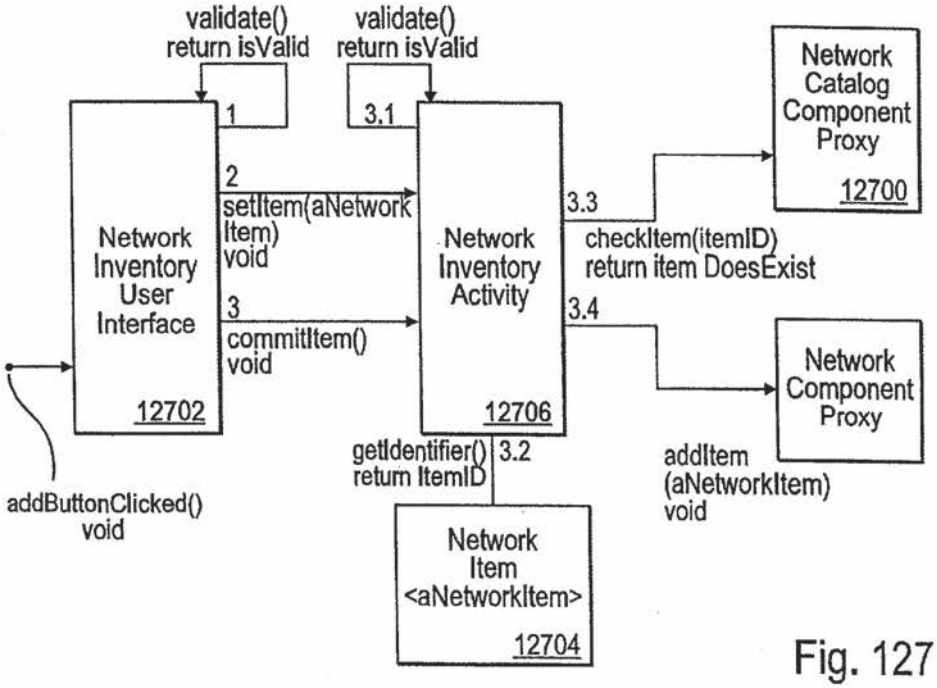


Fig. 127

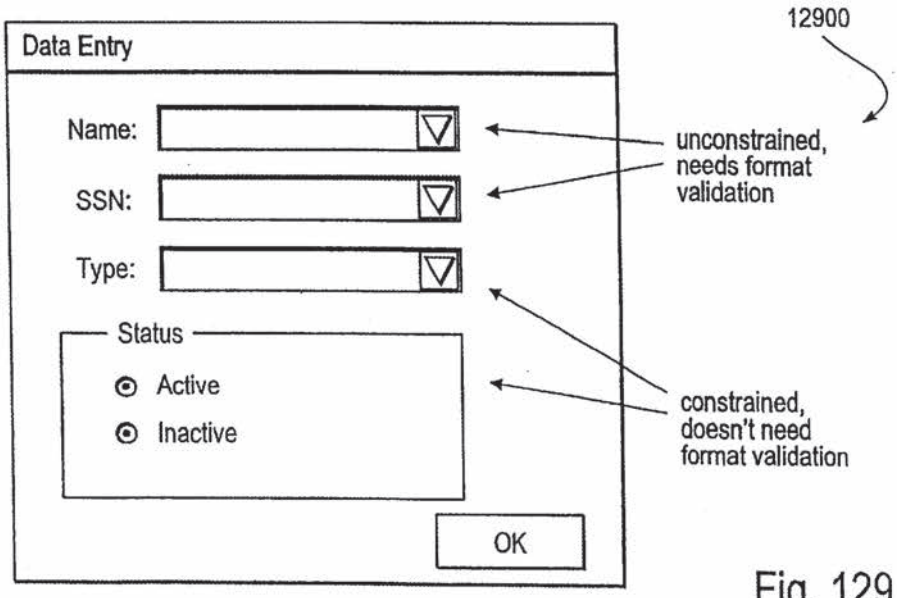


Fig. 129

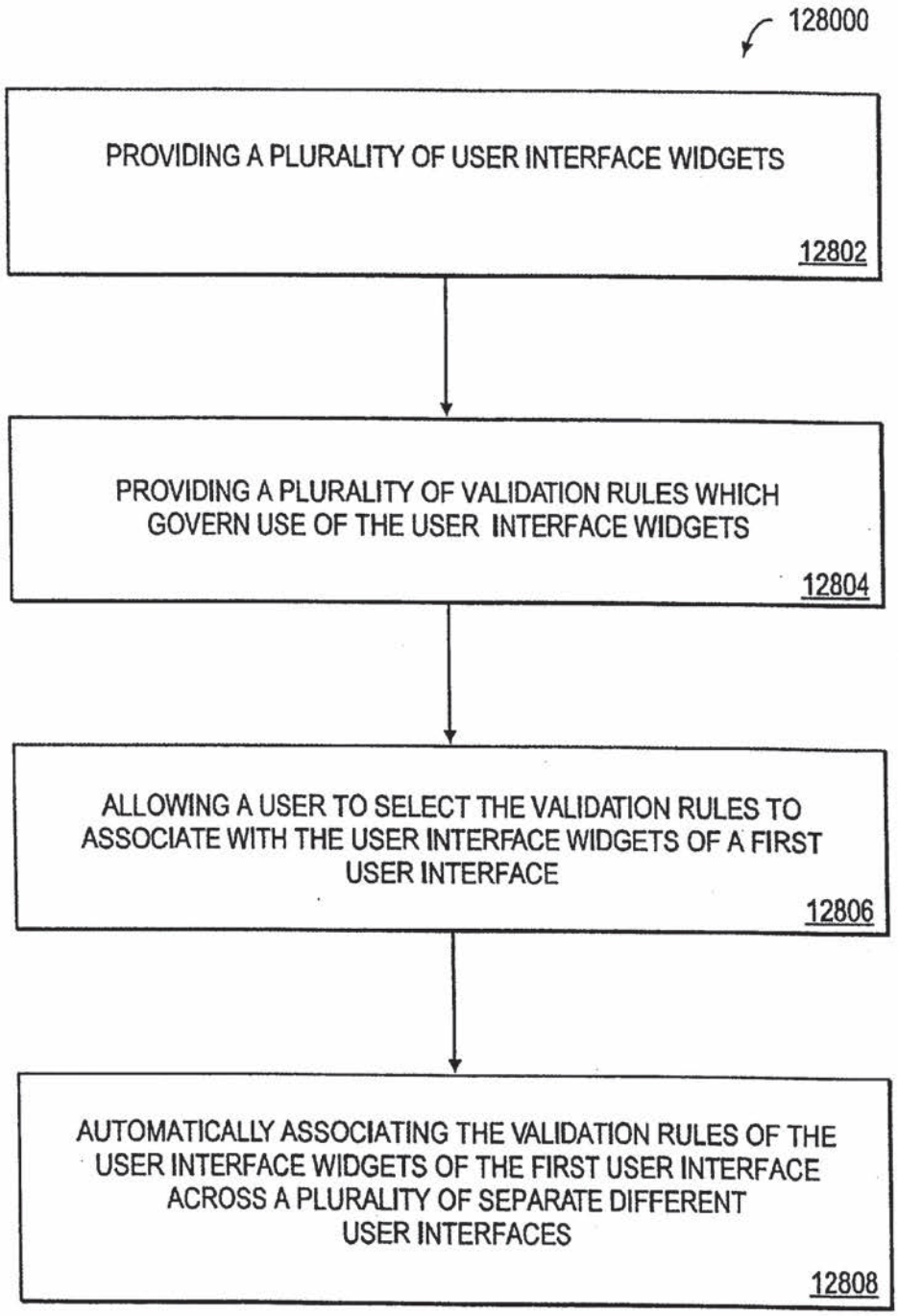

128000
PROVIDING A PLURALITY OF USER INTERFACE WIDGETS
12802
PROVIDING A PLURALITY OF VALIDATION RULES WHICH GOVERN USE OF THE USER INTERFACE WIDGETS
12804
ALLOWING A USER TO SELECT THE VALIDATION RULES TO ASSOCIATE WITH THE USER INTERFACE WIDGETS OF A FIRST USER INTERFACE
12806
AUTOMATICALLY ASSOCIATING THE VALIDATION RULES OF THE USER INTERFACE WIDGETS OF THE FIRST USER INTERFACE ACROSS A PLURALITY OF SEPARATE DIFFERENT USER INTERFACES
12808

Fig. 128

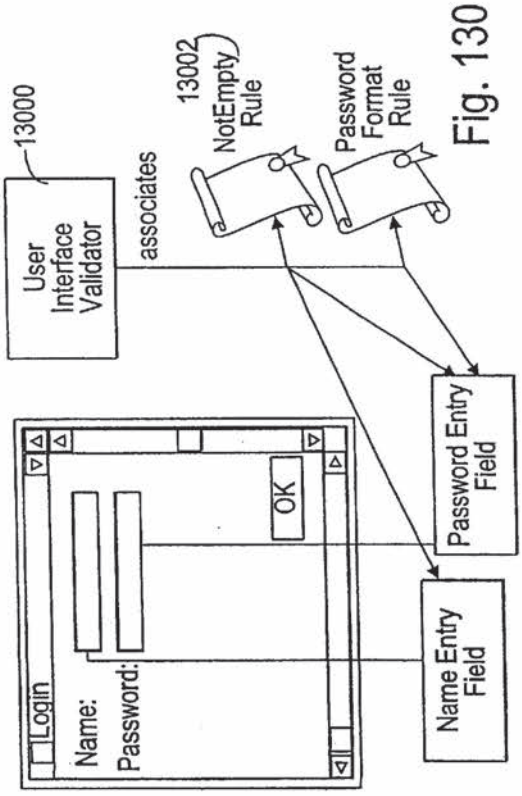


Fig. 130

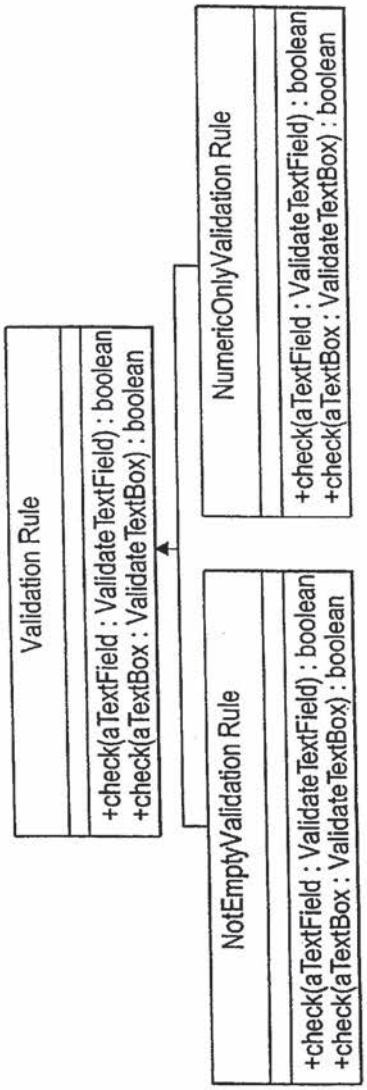


Fig. 131

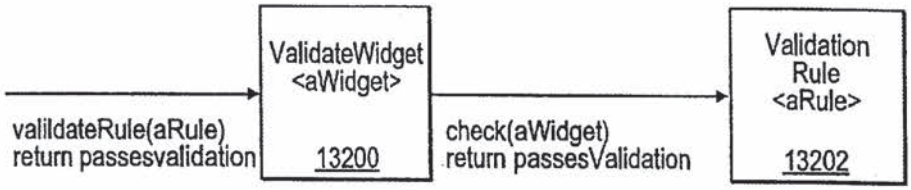


Fig. 132

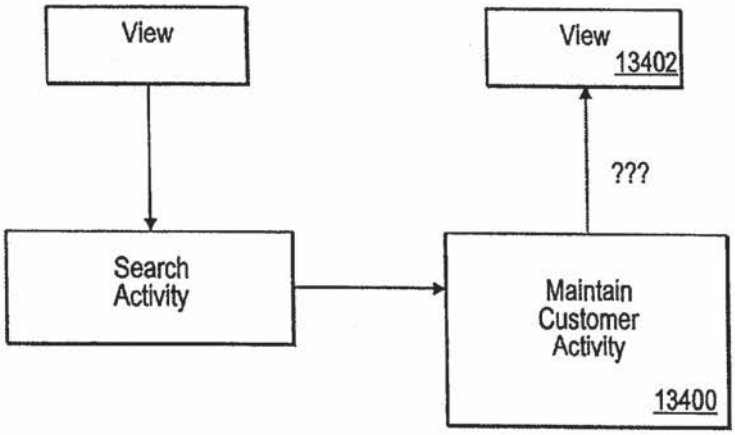


Fig. 134

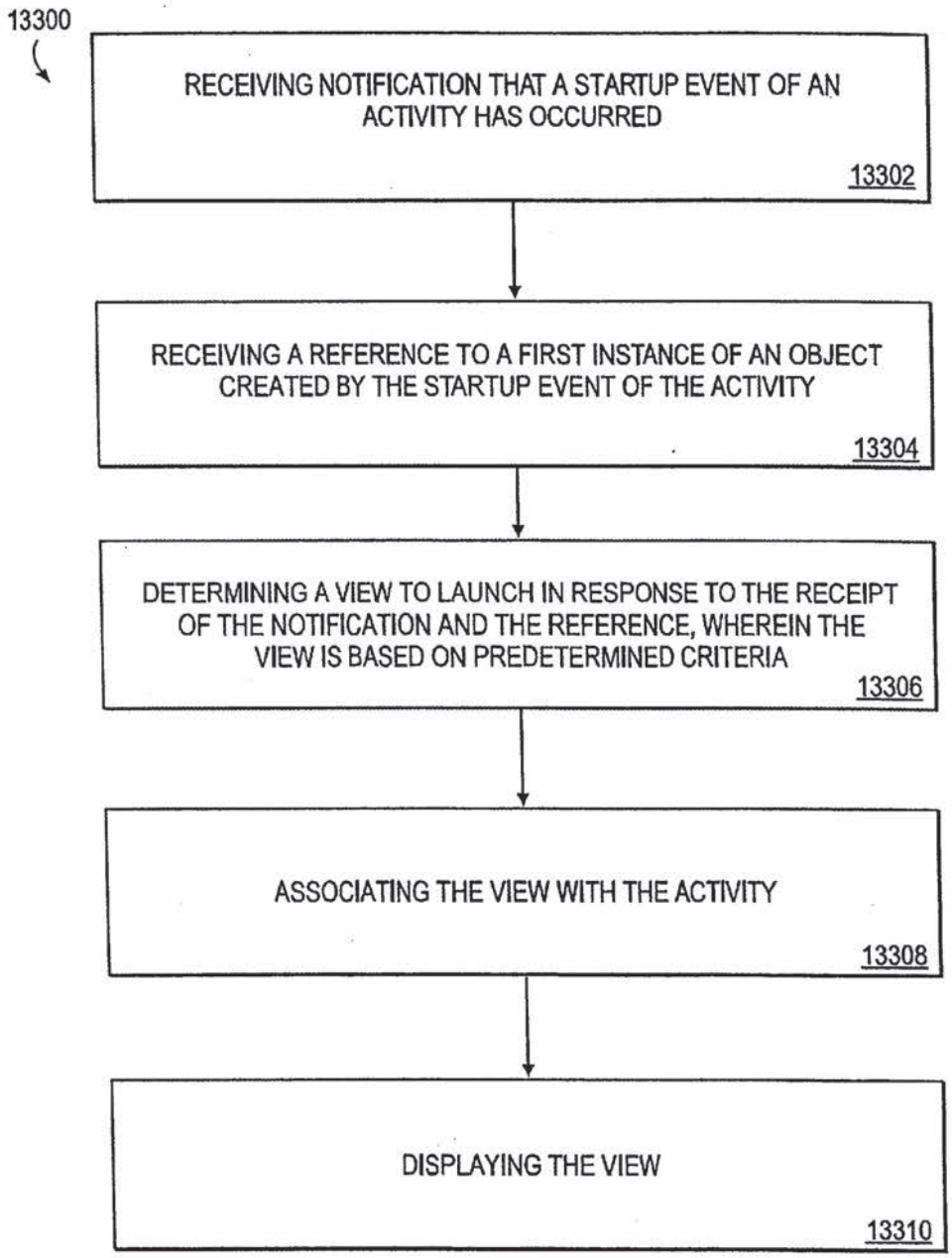

13300
RECEIVING NOTIFICATION THAT A STARTUP EVENT OF AN ACTIVITY HAS OCCURRED
13302
RECEIVING A REFERENCE TO A FIRST INSTANCE OF AN OBJECT CREATED BY THE STARTUP EVENT OF THE ACTIVITY
13304
DETERMINING A VIEW TO LAUNCH IN RESPONSE TO THE RECEIPT OF THE NOTIFICATION AND THE REFERENCE, WHEREIN THE VIEW IS BASED ON PREDETERMINED CRITERIA
13306
ASSOCIATING THE VIEW WITH THE ACTIVITY
13308
DISPLAYING THE VIEW
13310

Fig. 133

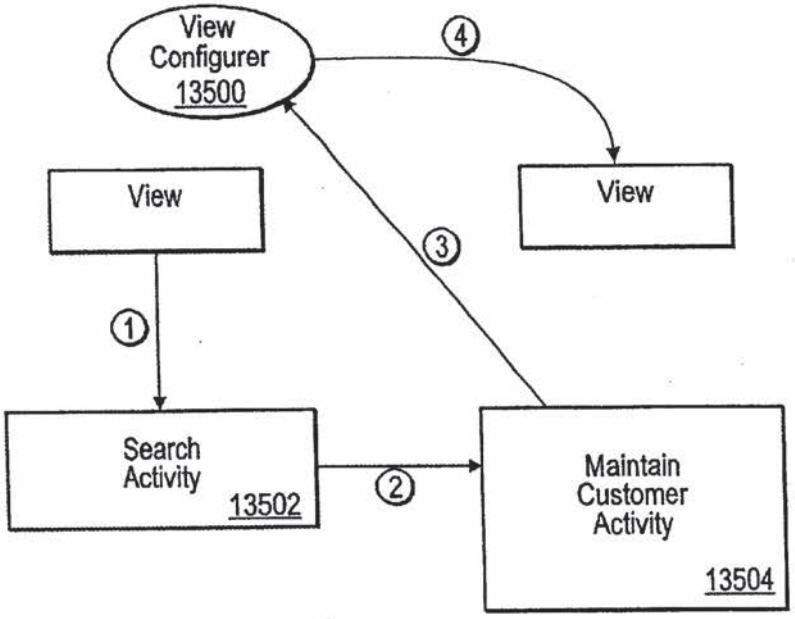


Fig. 135

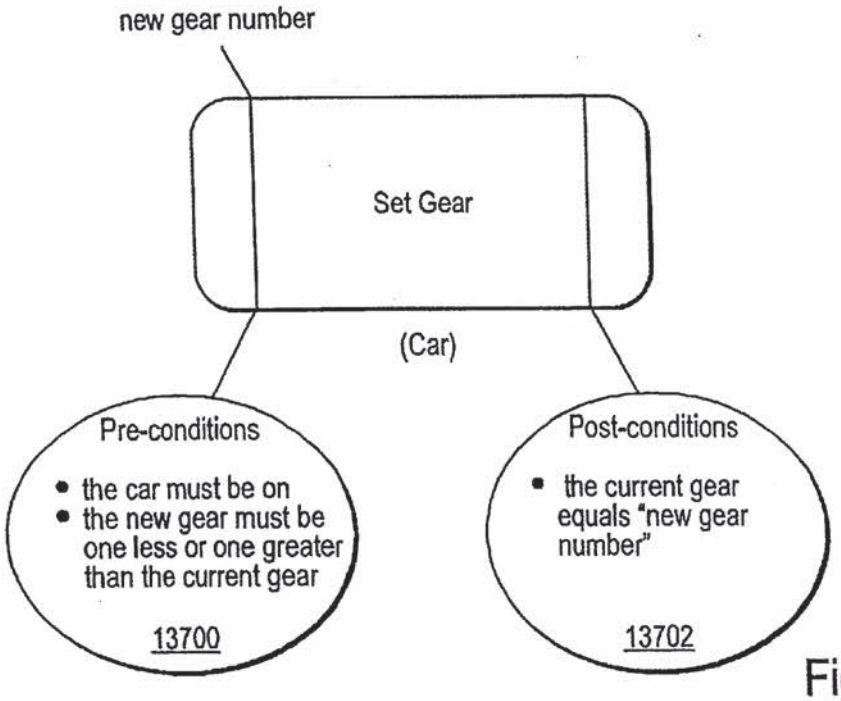


Fig. 137

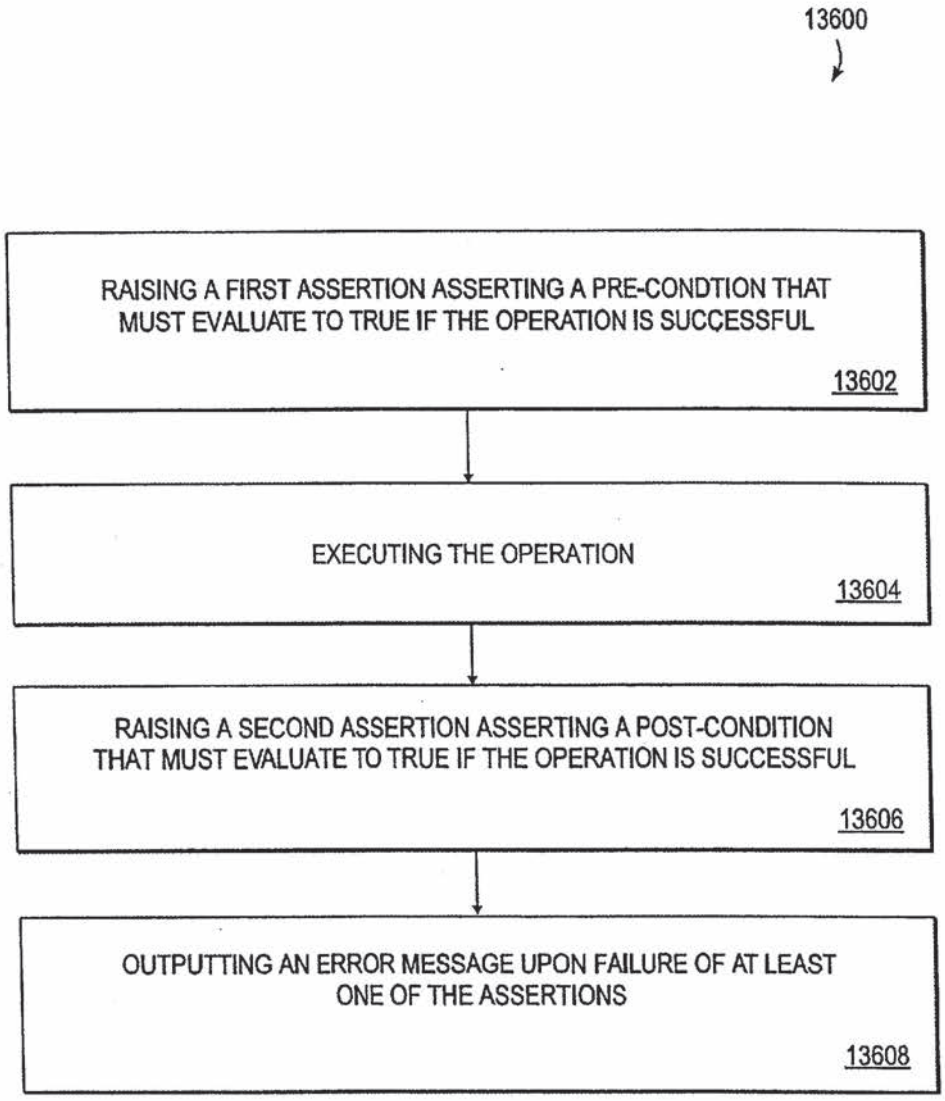

13600
RAISING A FIRST ASSERTION ASSERTING A PRE-CONDTION THAT MUST EVALUATE TO TRUE IF THE OPERATION IS SUCCESSFUL
13602
EXECUTING THE OPERATION
13604
RAISING A SECOND ASSERTION ASSERTING A POST-CONDITION THAT MUST EVALUATE TO TRUE IF THE OPERATION IS SUCCESSFUL
13606
OUTPUTTING AN ERROR MESSAGE UPON FAILURE OF AT LEAST ONE OF THE ASSERTIONS
13608

Fig. 136

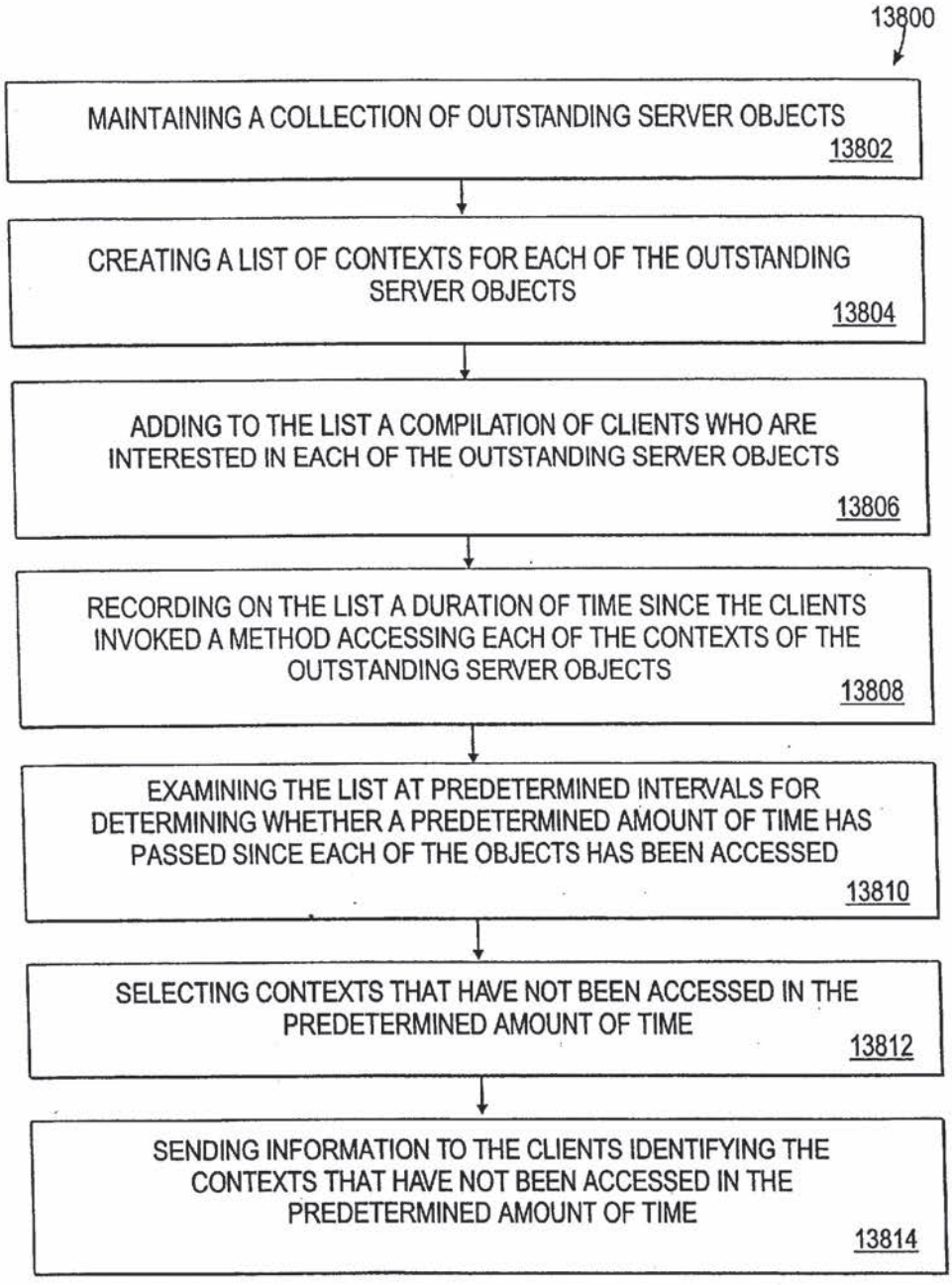

13800
MAINTAINING A COLLECTION OF OUTSTANDING SERVER OBJECTS
13802
CREATING A LIST OF CONTEXTS FOR EACH OF THE OUTSTANDING SERVER OBJECTS
13804
ADDING TO THE LIST A COMPILATION OF CLIENTS WHO ARE INTERESTED IN EACH OF THE OUTSTANDING SERVER OBJECTS
13806
RECORDING ON THE LIST A DURATION OF TIME SINCE THE CLIENTS INVOKED A METHOD ACCESSING EACH OF THE CONTEXTS OF THE OUTSTANDING SERVER OBJECTS
13808
EXAMINING THE LIST AT PREDETERMINED INTERVALS FOR DETERMINING WHETHER A PREDETERMINED AMOUNT OF TIME HAS PASSED SINCE EACH OF THE OBJECTS HAS BEEN ACCESSED
13810
SELECTING CONTEXTS THAT HAVE NOT BEEN ACCESSED IN THE PREDETERMINED AMOUNT OF TIME
13812
SENDING INFORMATION TO THE CLIENTS IDENTIFYING THE CONTEXTS THAT HAVE NOT BEEN ACCESSED IN THE PREDETERMINED AMOUNT OF TIME
13814

Fig. 138

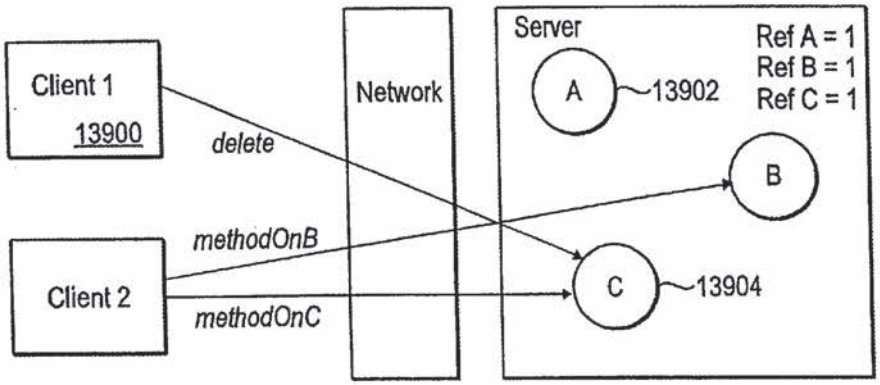


Fig. 139

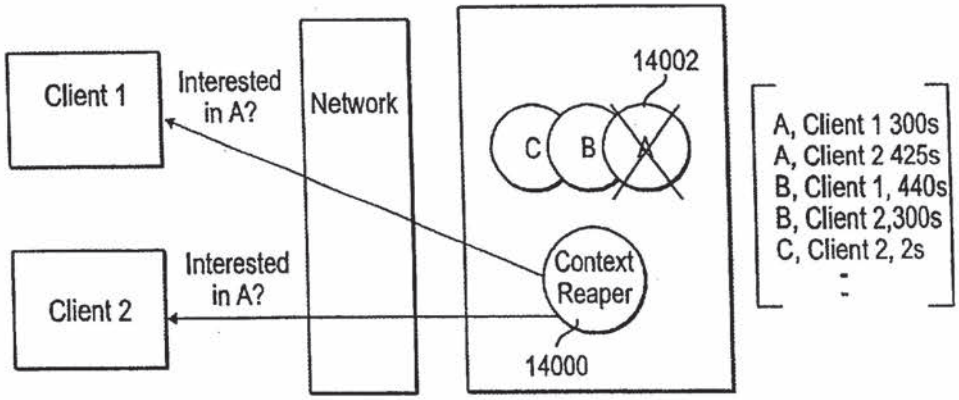


Fig. 140

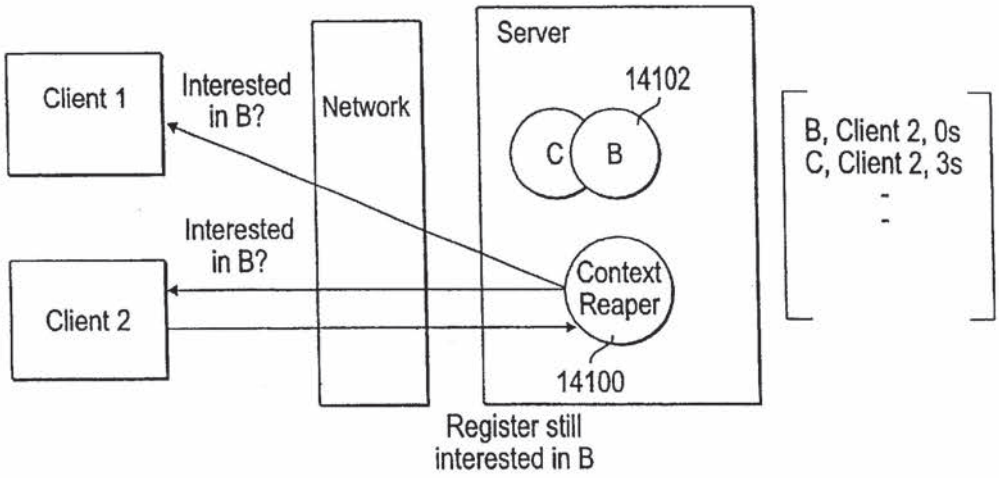


Fig. 141

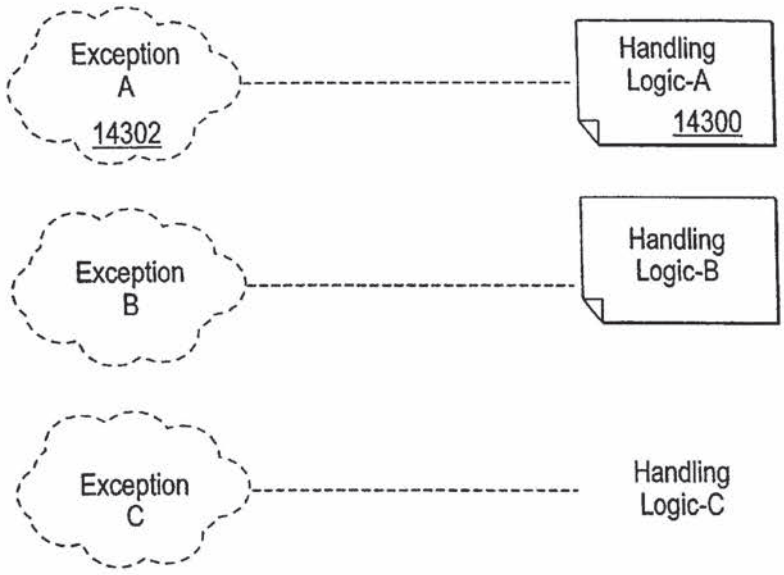


Fig. 143

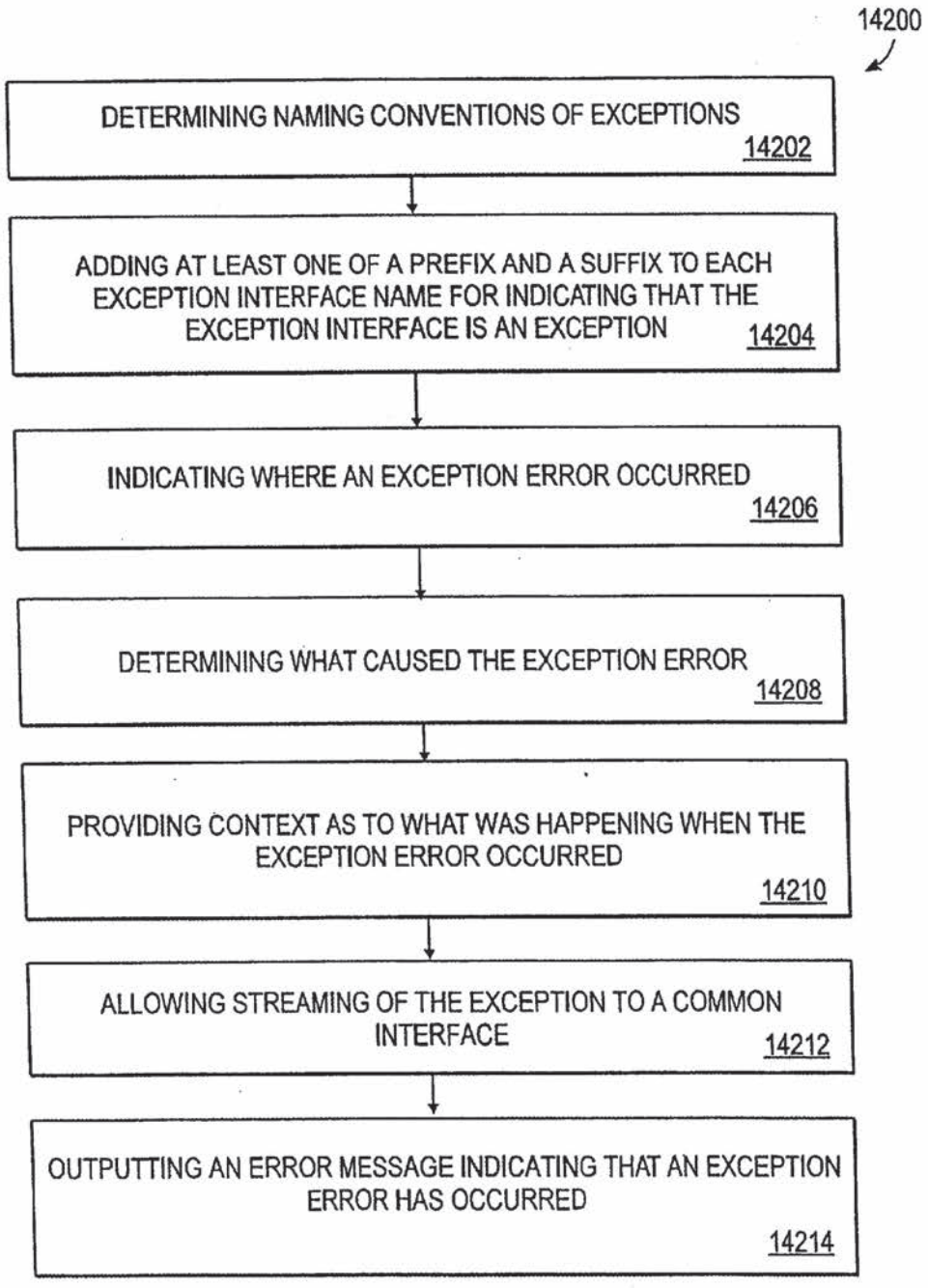

14200
DETERMINING NAMING CONVENTIONS OF EXCEPTIONS
14202
ADDING AT LEAST ONE OF A PREFIX AND A SUFFIX TO EACH EXCEPTION INTERFACE NAME FOR INDICATING THAT THE EXCEPTION INTERFACE IS AN EXCEPTION
14204
INDICATING WHERE AN EXCEPTION ERROR OCCURRED
14206
DETERMINING WHAT CAUSED THE EXCEPTION ERROR
14208
PROVIDING CONTEXT AS TO WHAT WAS HAPPENING WHEN THE EXCEPTION ERROR OCCURRED
14210
ALLOWING STREAMING OF THE EXCEPTION TO A COMMON INTERFACE
14212
OUTPUTTING AN ERROR MESSAGE INDICATING THAT AN EXCEPTION ERROR HAS OCCURRED
14214

Fig. 142

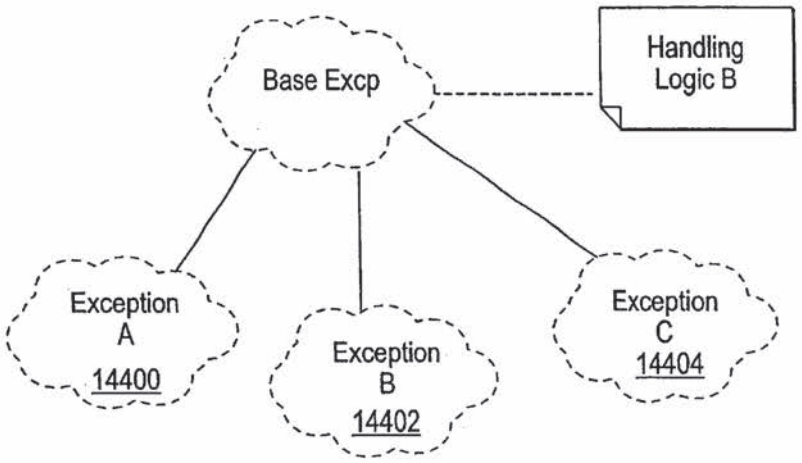


Fig. 144

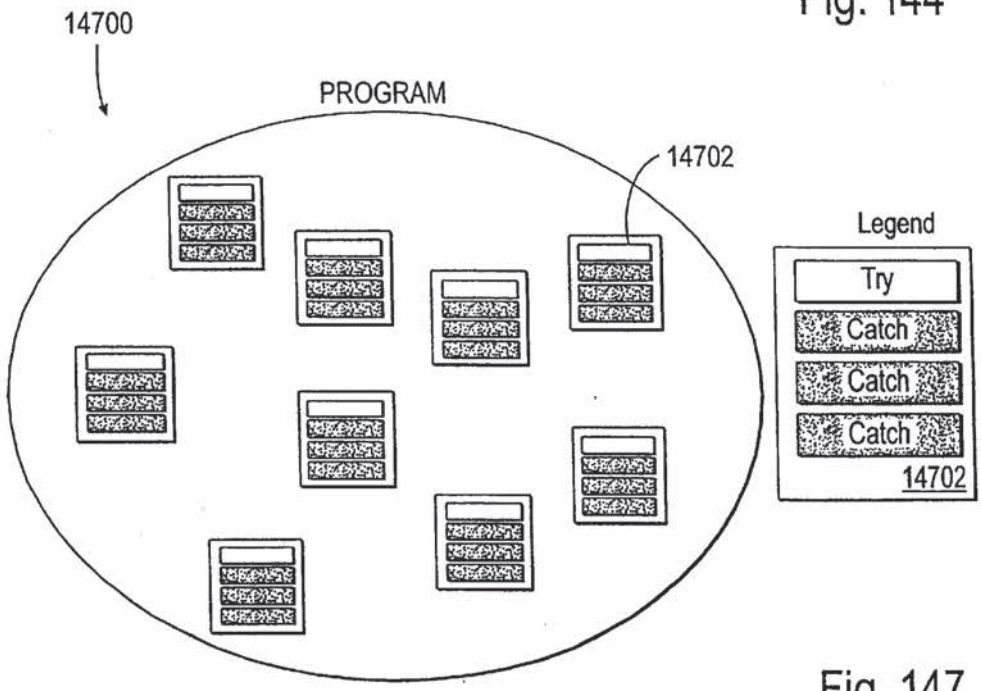


Fig. 147

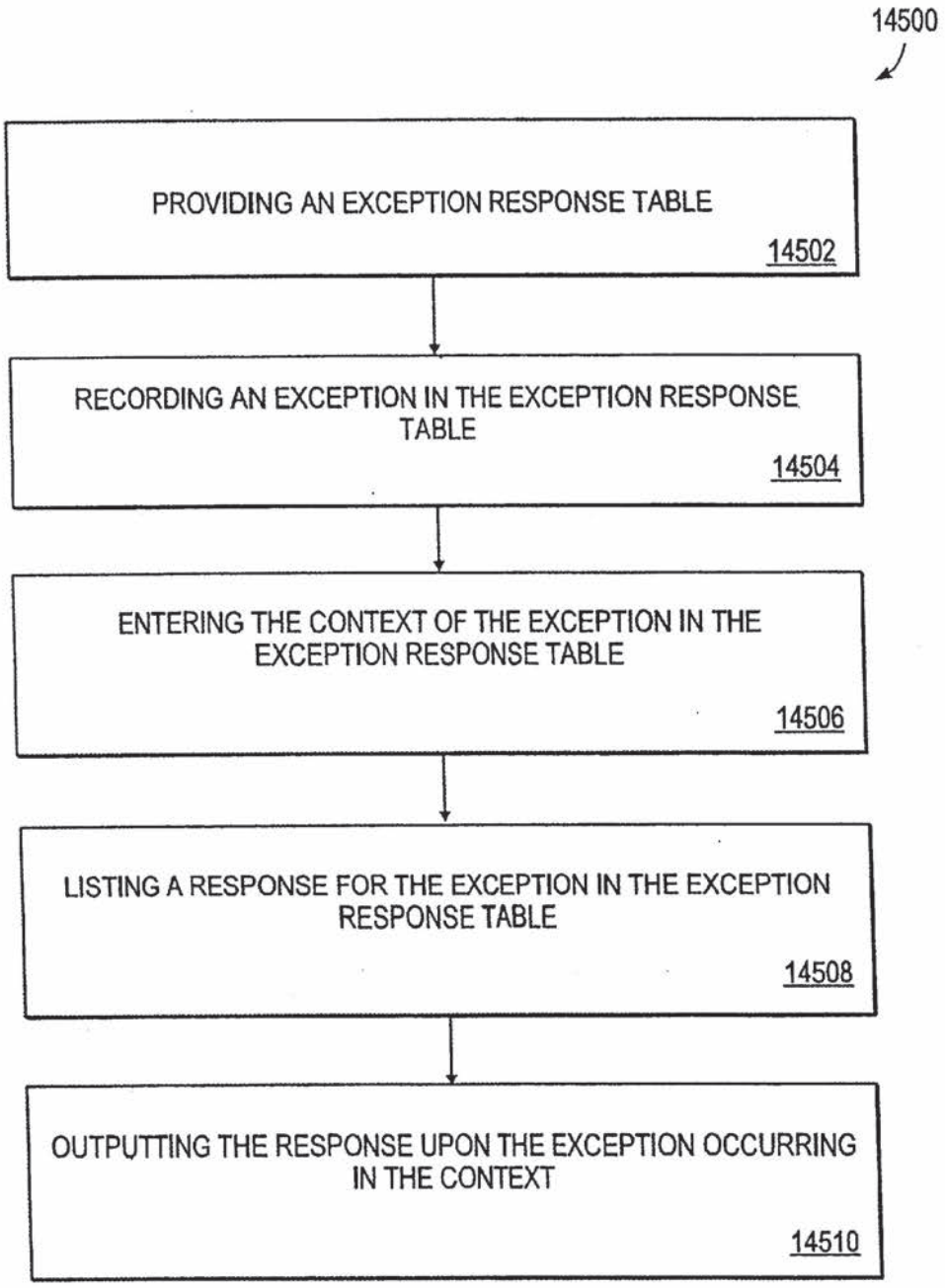

14500
PROVIDING AN EXCEPTION RESPONSE TABLE
14502
RECORDING AN EXCEPTION IN THE EXCEPTION RESPONSE TABLE
14504
ENTERING THE CONTEXT OF THE EXCEPTION IN THE EXCEPTION RESPONSE TABLE
14506
LISTING A RESPONSE FOR THE EXCEPTION IN THE EXCEPTION RESPONSE TABLE
14508
OUTPUTTING THE RESPONSE UPON THE EXCEPTION OCCURRING IN THE CONTEXT
14510

Fig. 145

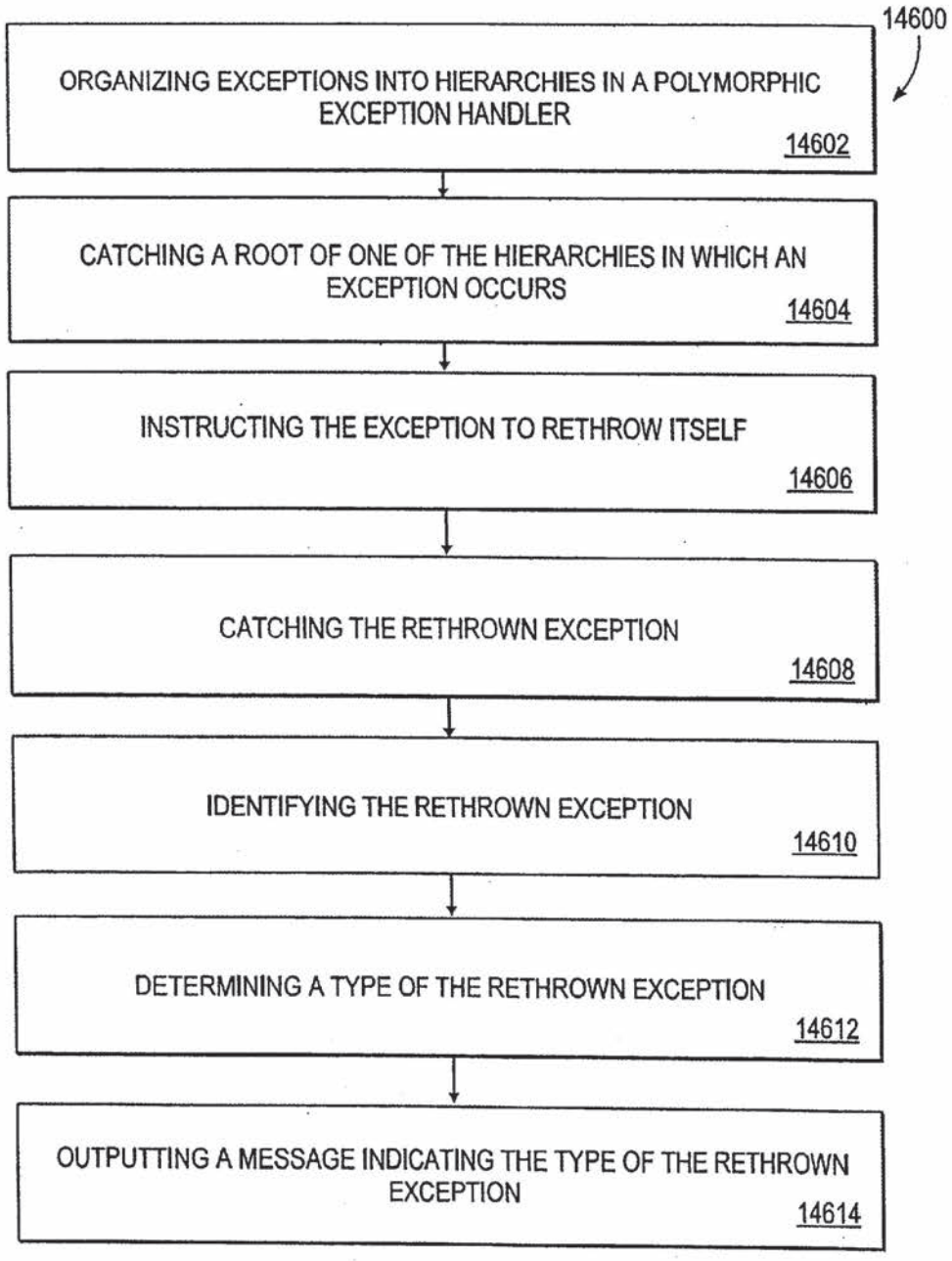

14600
ORGANIZING EXCEPTIONS INTO HIERARCHIES IN A POLYMORPHIC EXCEPTION HANDLER
14602
CATCHING A ROOT OF ONE OF THE HIERARCHIES IN WHICH AN EXCEPTION OCCURS
14604
INSTRUCTING THE EXCEPTION TO RETHROW ITSELF
14606
CATCHING THE RETHROWN EXCEPTION
14608
IDENTIFYING THE RETHROWN EXCEPTION
14610
DETERMINING A TYPE OF THE RETHROWN EXCEPTION
14612
OUTPUTTING A MESSAGE INDICATING THE TYPE OF THE RETHROWN EXCEPTION
14614

Fig. 146

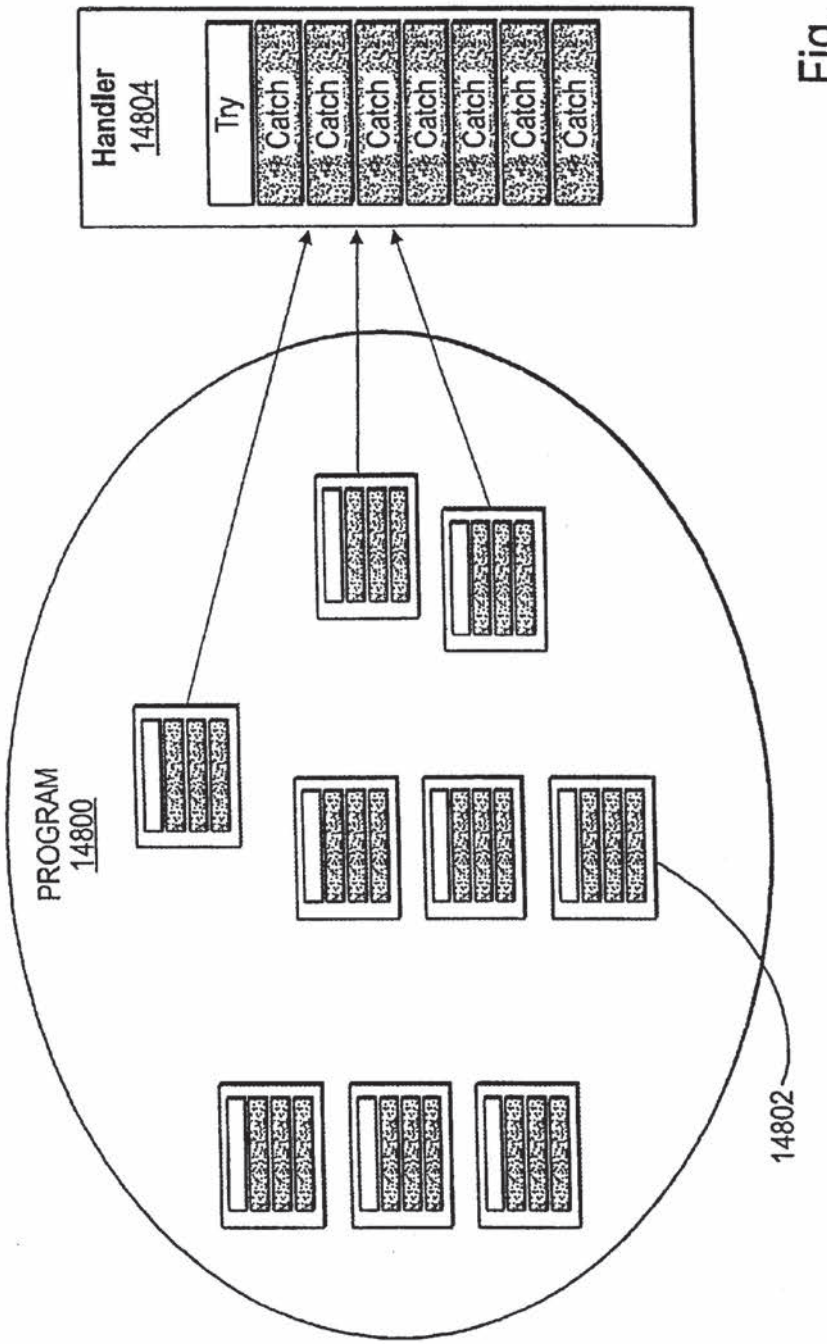

Handler
14804
Try
Catch
Catch
Catch
Catch
Catch
Catch
Catch
PROGRAM
14800
14802

Fig. 148

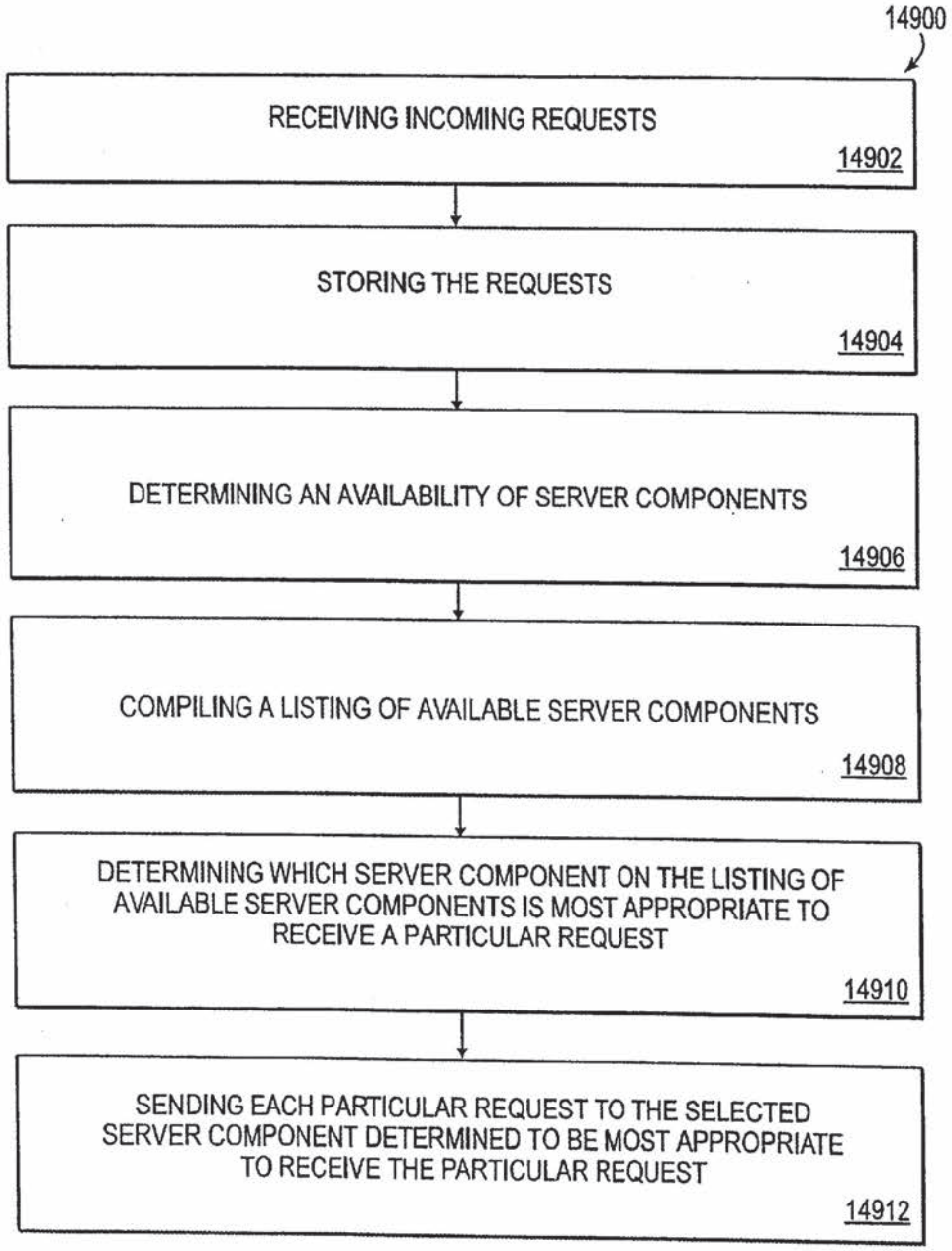

14900
RECEIVING INCOMING REQUESTS
14902
STORING THE REQUESTS
14904
DETERMINING AN AVAILABILITY OF SERVER COMPONENTS
14906
COMPILING A LISTING OF AVAILABLE SERVER COMPONENTS
14908
DETERMINING WHICH SERVER COMPONENT ON THE LISTING OF AVAILABLE SERVER COMPONENTS IS MOST APPROPRIATE TO RECEIVE A PARTICULAR REQUEST
14910
SENDING EACH PARTICULAR REQUEST TO THE SELECTED SERVER COMPONENT DETERMINED TO BE MOST APPROPRIATE TO RECEIVE THE PARTICULAR REQUEST
14912

Fig. 149

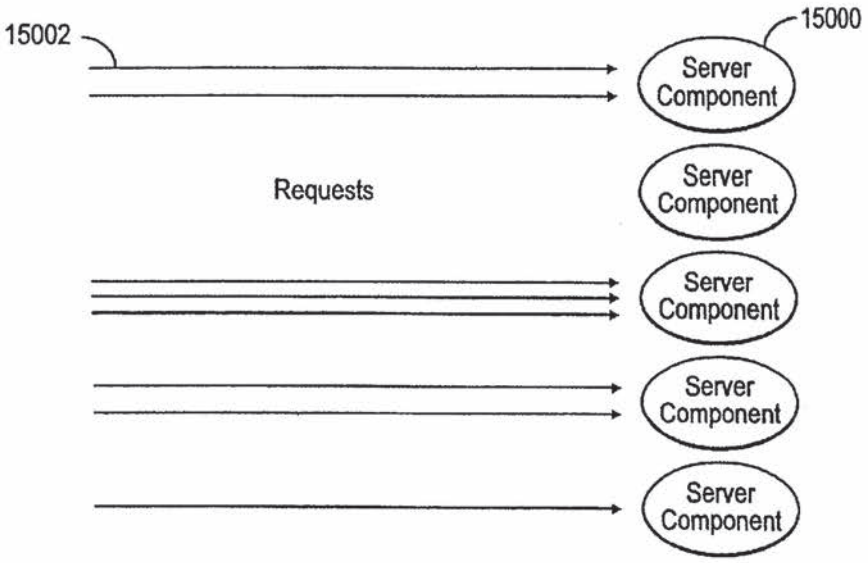


Fig. 150

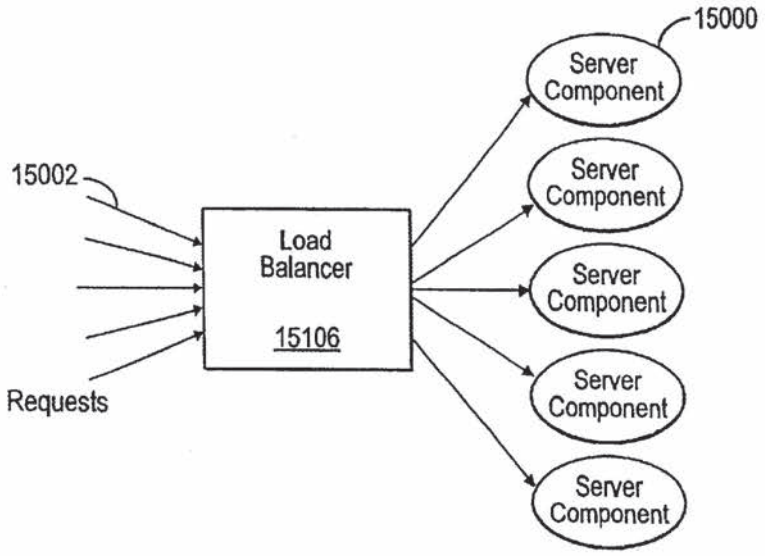


Fig. 151

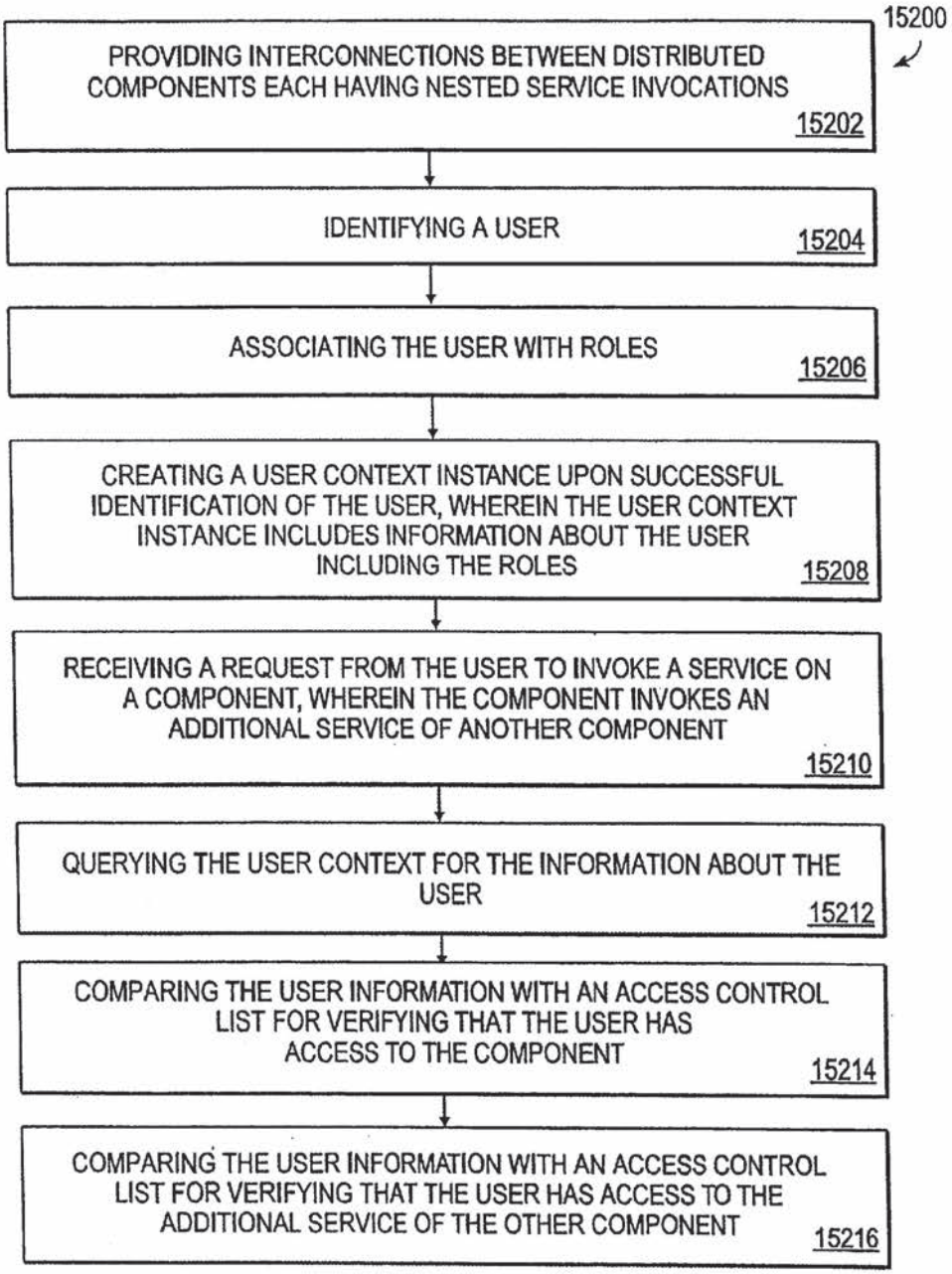

15200
PROVIDING INTERCONNECTIONS BETWEEN DISTRIBUTED COMPONENTS EACH HAVING NESTED SERVICE INVOCATIONS
15202
IDENTIFYING A USER
15204
ASSOCIATING THE USER WITH ROLES
15206
CREATING A USER CONTEXT INSTANCE UPON SUCCESSFUL IDENTIFICATION OF THE USER, WHEREIN THE USER CONTEXT INSTANCE INCLUDES INFORMATION ABOUT THE USER INCLUDING THE ROLES
15208
RECEIVING A REQUEST FROM THE USER TO INVOKE A SERVICE ON A COMPONENT, WHEREIN THE COMPONENT INVOKES AN ADDITIONAL SERVICE OF ANOTHER COMPONENT
15210
QUERYING THE USER CONTEXT FOR THE INFORMATION ABOUT THE USER
15212
COMPARING THE USER INFORMATION WITH AN ACCESS CONTROL LIST FOR VERIFYING THAT THE USER HAS ACCESS TO THE COMPONENT
15214
COMPARING THE USER INFORMATION WITH AN ACCESS CONTROL LIST FOR VERIFYING THAT THE USER HAS ACCESS TO THE ADDITIONAL SERVICE OF THE OTHER COMPONENT
15216

Fig. 152

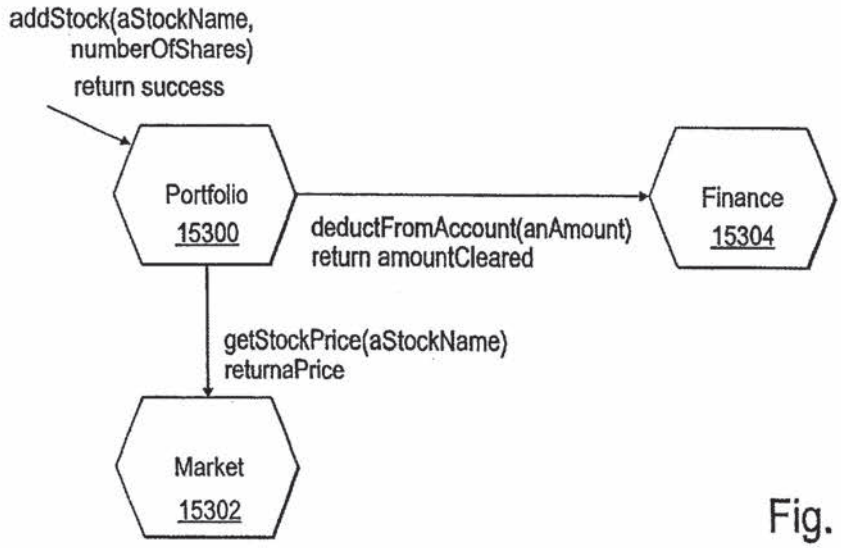


Fig. 153

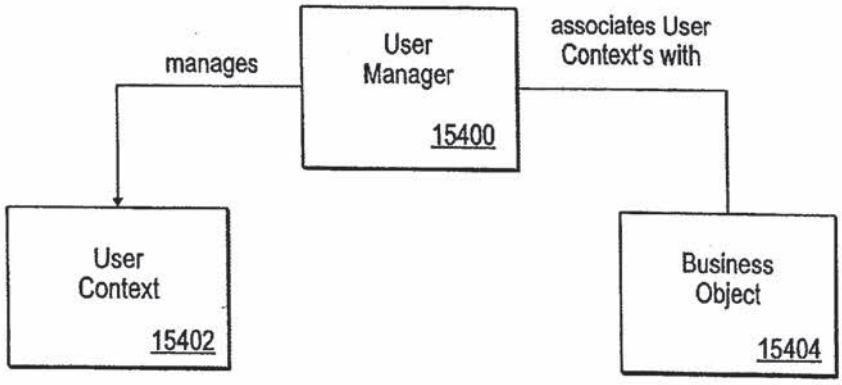


Fig. 154

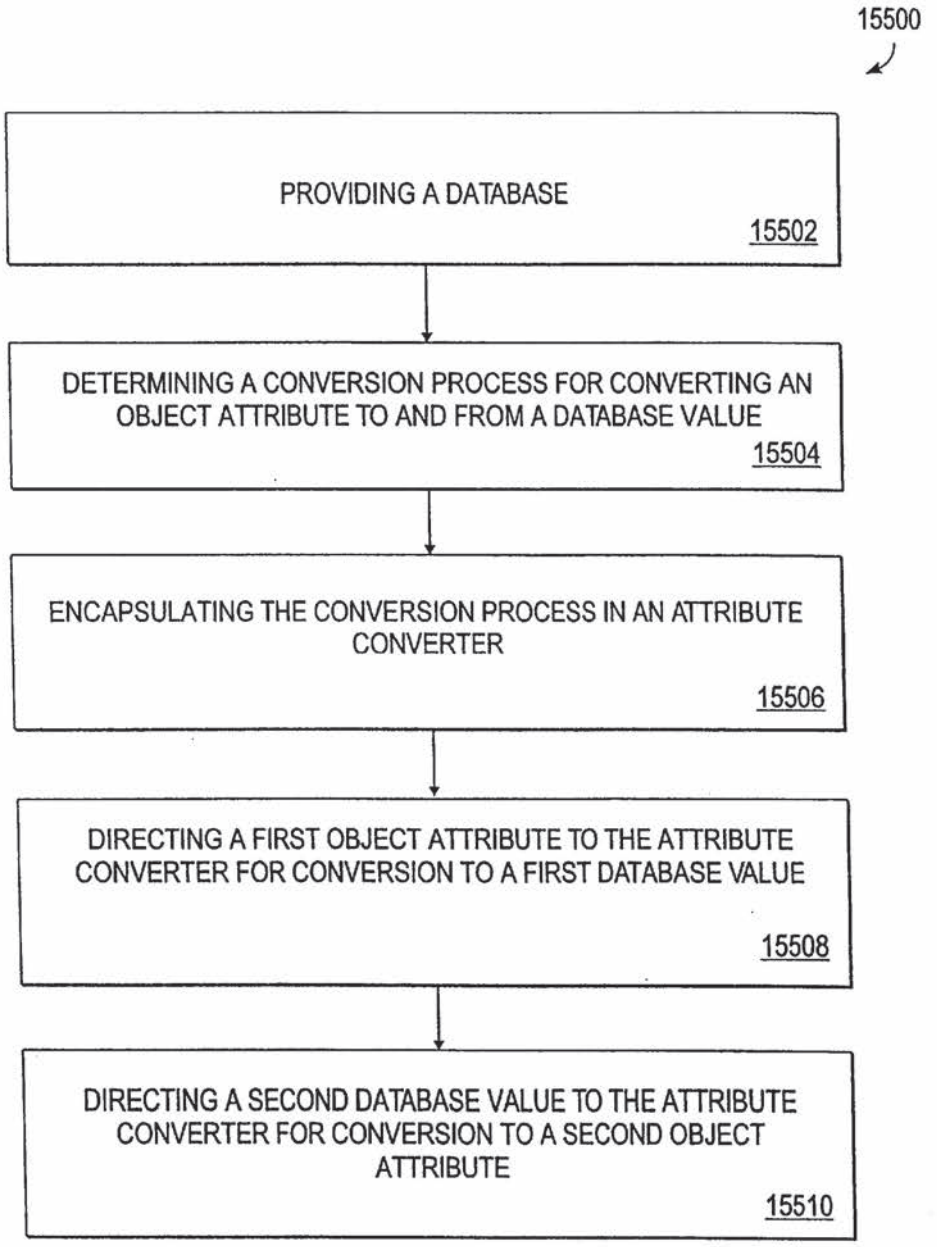

15500
PROVIDING A DATABASE
15502
DETERMINING A CONVERSION PROCESS FOR CONVERTING AN OBJECT ATTRIBUTE TO AND FROM A DATABASE VALUE
15504
ENCAPSULATING THE CONVERSION PROCESS IN AN ATTRIBUTE CONVERTER
15506
DIRECTING A FIRST OBJECT ATTRIBUTE TO THE ATTRIBUTE CONVERTER FOR CONVERSION TO A FIRST DATABASE VALUE
15508
DIRECTING A SECOND DATABASE VALUE TO THE ATTRIBUTE CONVERTER FOR CONVERSION TO A SECOND OBJECT ATTRIBUTE
15510

Fig. 155

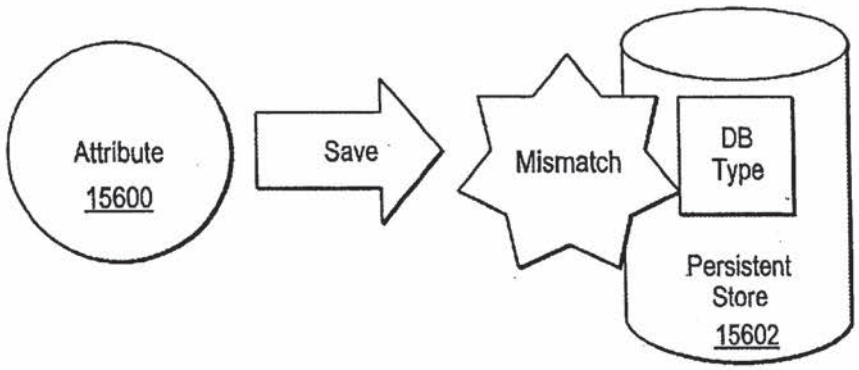


Fig. 156

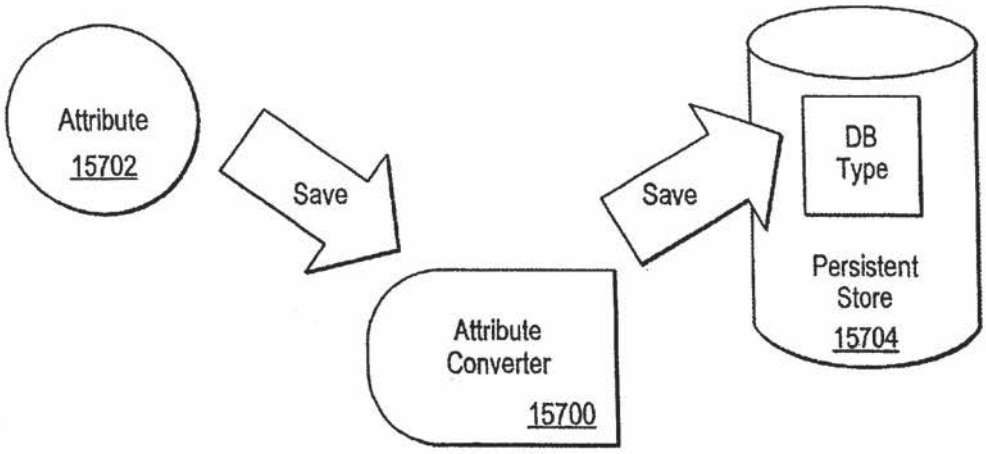


Fig. 157

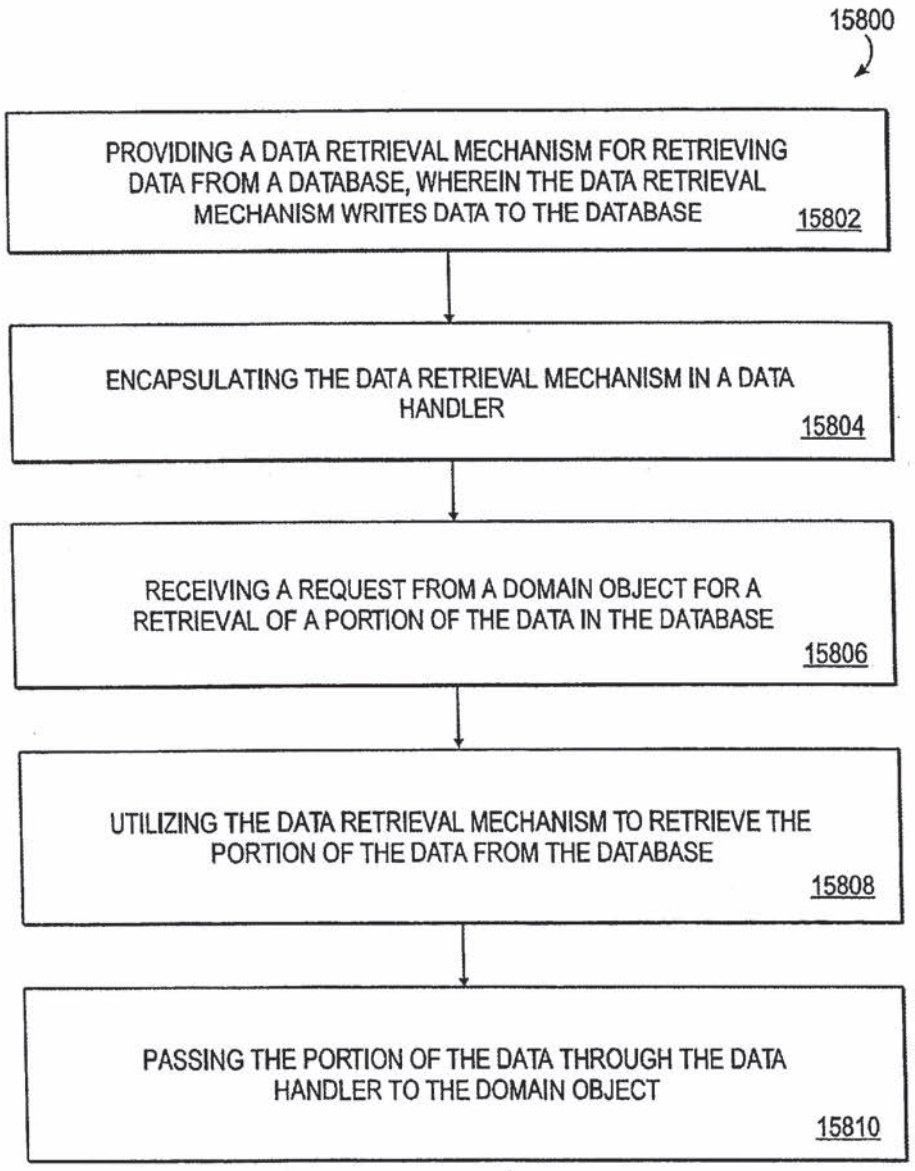

15800
PROVIDING A DATA RETRIEVAL MECHANISM FOR RETRIEVING DATA FROM A DATABASE, WHEREIN THE DATA RETRIEVAL MECHANISM WRITES DATA TO THE DATABASE
15802
ENCAPSULATING THE DATA RETRIEVAL MECHANISM IN A DATA HANDLER
15804
RECEIVING A REQUEST FROM A DOMAIN OBJECT FOR A RETRIEVAL OF A PORTION OF THE DATA IN THE DATABASE
15806
UTILIZING THE DATA RETRIEVAL MECHANISM TO RETRIEVE THE PORTION OF THE DATA FROM THE DATABASE
15808
PASSING THE PORTION OF THE DATA THROUGH THE DATA HANDLER TO THE DOMAIN OBJECT
15810

Fig. 158

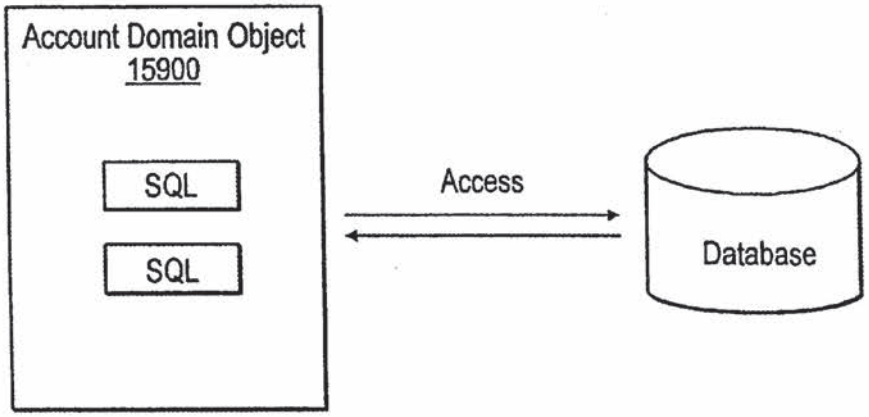


Fig. 159

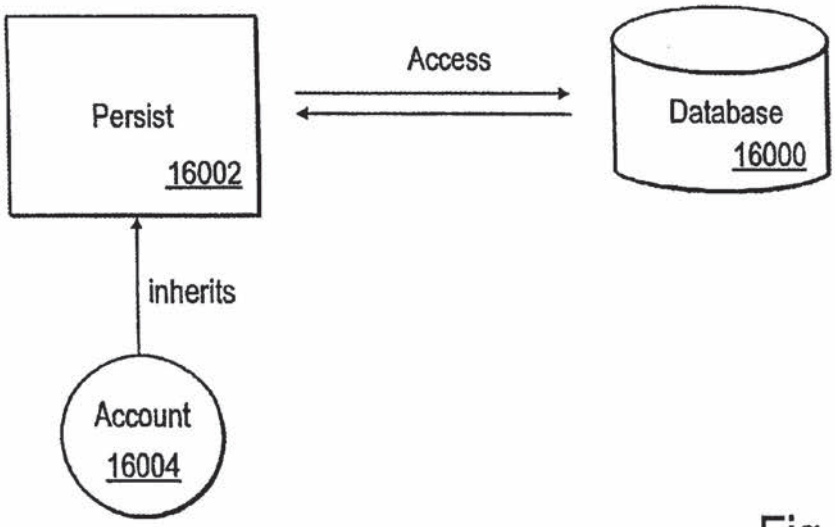


Fig. 160

Fig. 161

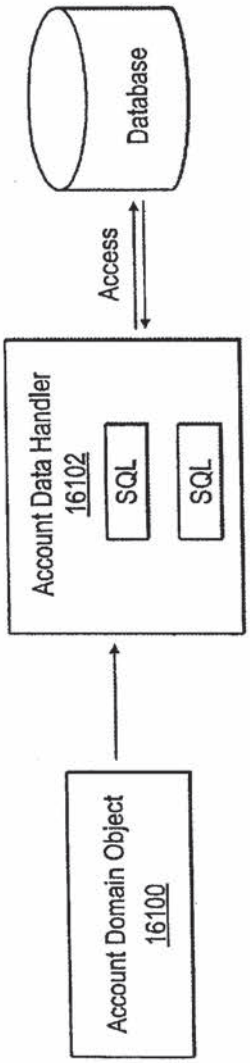
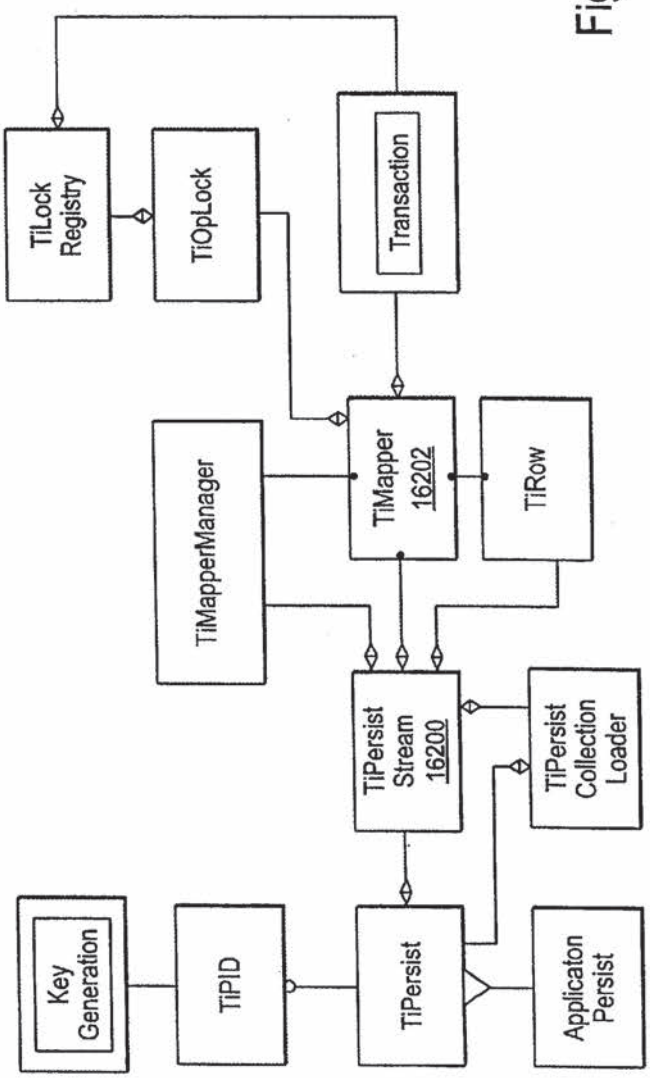


Fig. 162



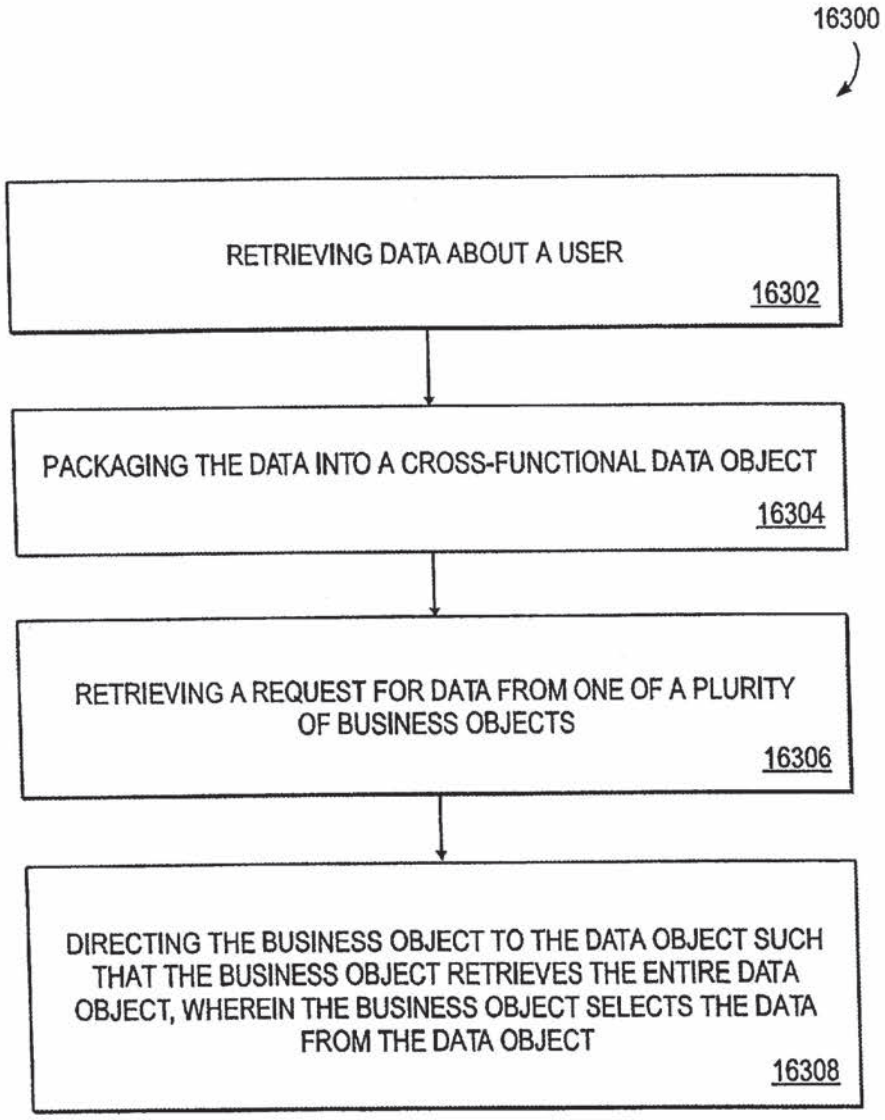

16300
RETRIEVING DATA ABOUT A USER
16302
PACKAGING THE DATA INTO A CROSS-FUNCTIONAL DATA OBJECT
16304
RETRIEVING A REQUEST FOR DATA FROM ONE OF A PLURITY OF BUSINESS OBJECTS
16306
DIRECTING THE BUSINESS OBJECT TO THE DATA OBJECT SUCH THAT THE BUSINESS OBJECT RETRIEVES THE ENTIRE DATA OBJECT, WHEREIN THE BUSINESS OBJECT SELECTS THE DATA FROM THE DATA OBJECT
16308

Fig. 163

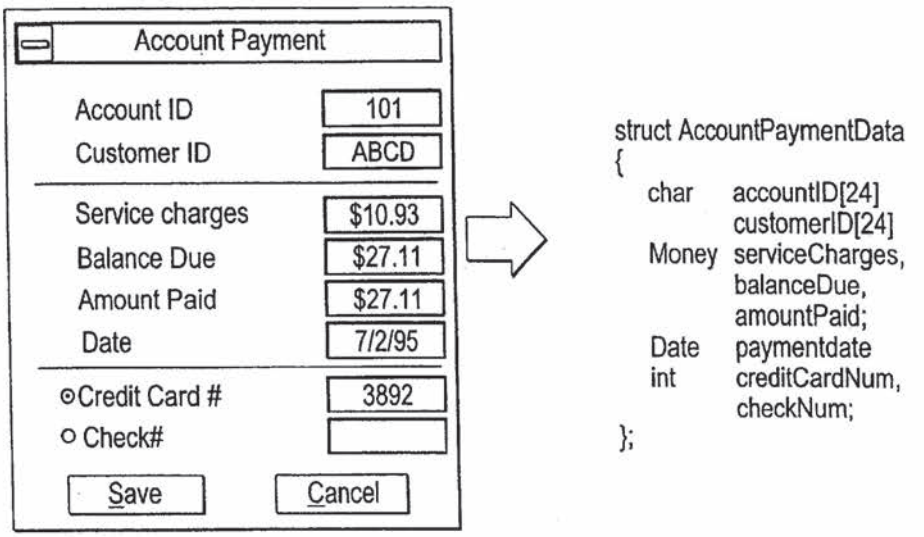


Fig. 164

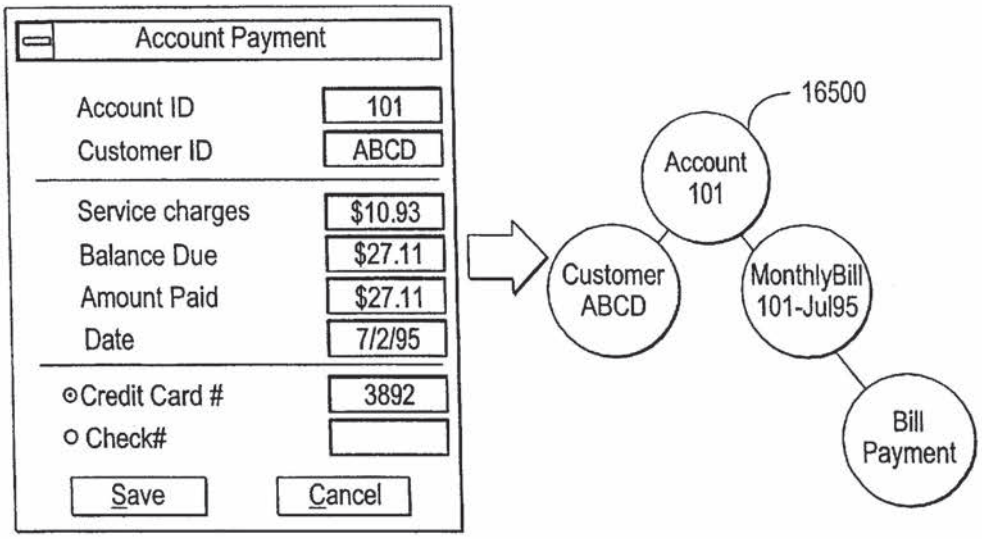


Fig. 165

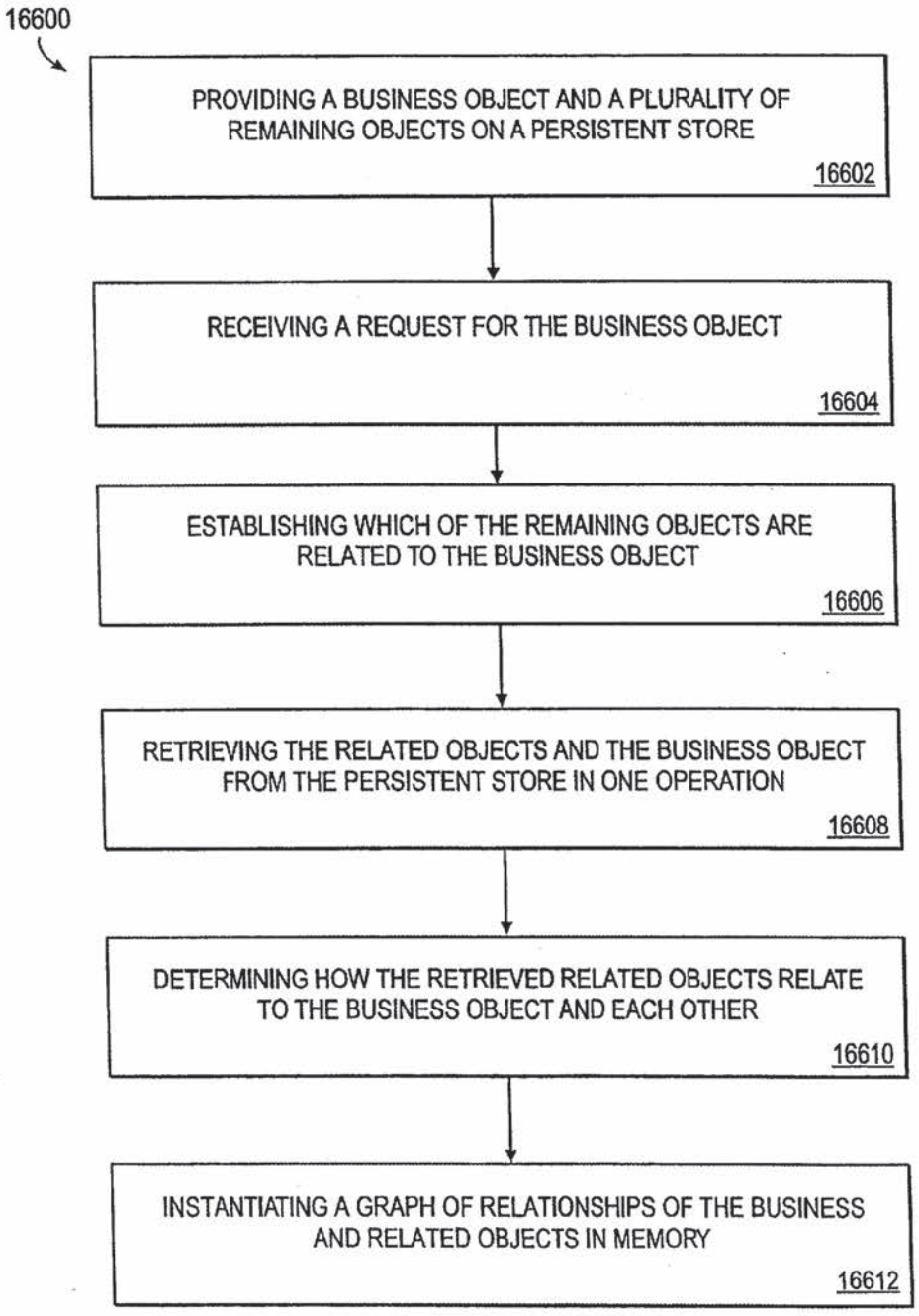

16600
PROVIDING A BUSINESS OBJECT AND A PLURALITY OF REMAINING OBJECTS ON A PERSISTENT STORE
16602
RECEIVING A REQUEST FOR THE BUSINESS OBJECT
16604
ESTABLISHING WHICH OF THE REMAINING OBJECTS ARE RELATED TO THE BUSINESS OBJECT
16606
RETRIEVING THE RELATED OBJECTS AND THE BUSINESS OBJECT FROM THE PERSISTENT STORE IN ONE OPERATION
16608
DETERMINING HOW THE RETRIEVED RELATED OBJECTS RELATE TO THE BUSINESS OBJECT AND EACH OTHER
16610
INSTANTIATING A GRAPH OF RELATIONSHIPS OF THE BUSINESS AND RELATED OBJECTS IN MEMORY
16612

Fig. 166

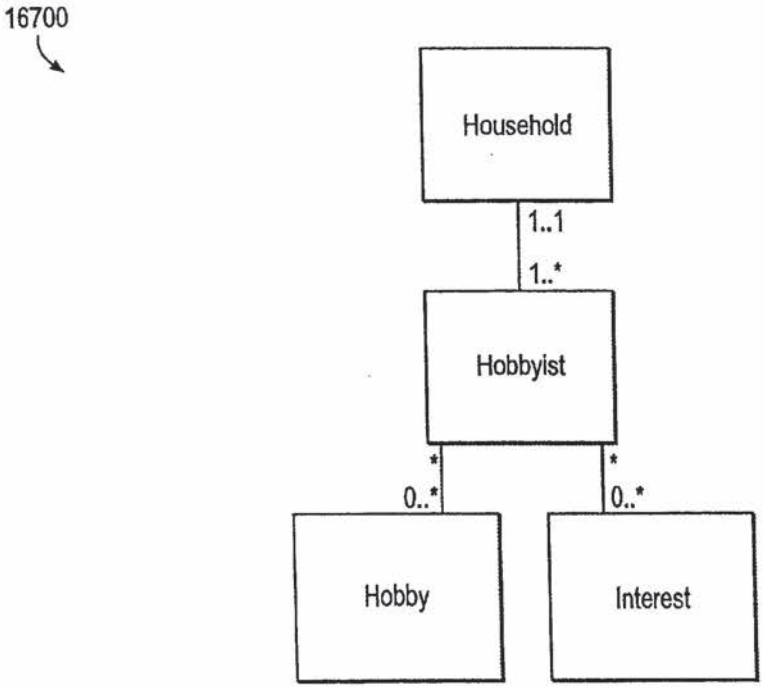


Fig. 167

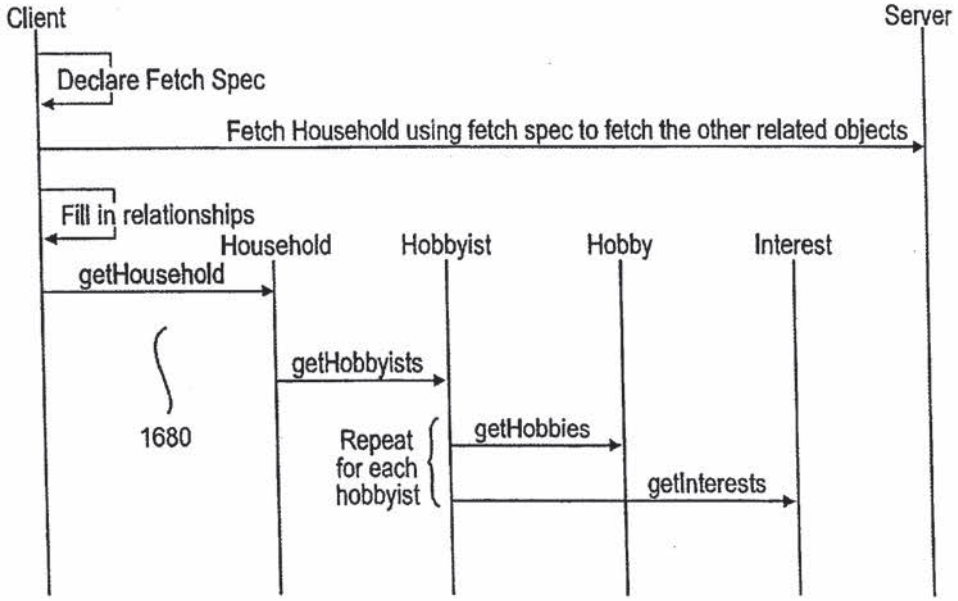


Fig. 168

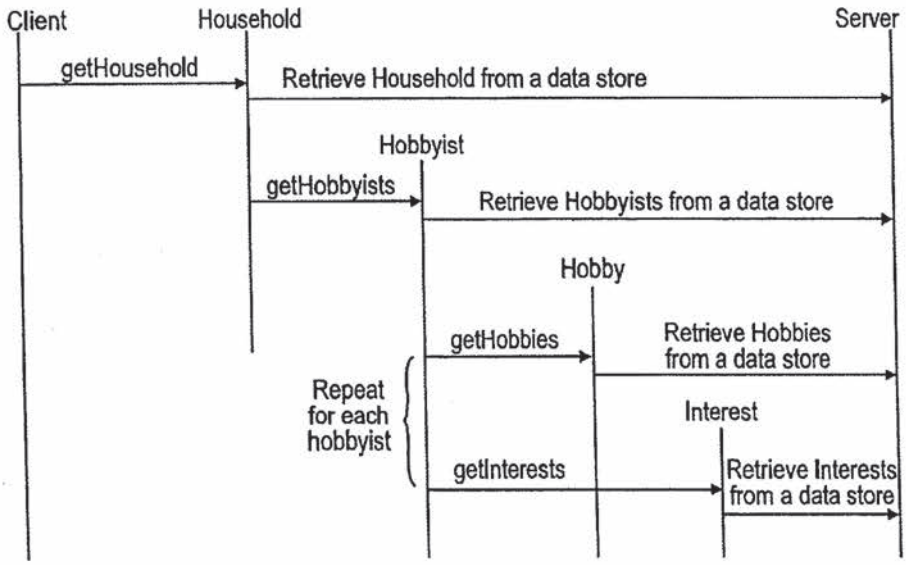


Fig. 169

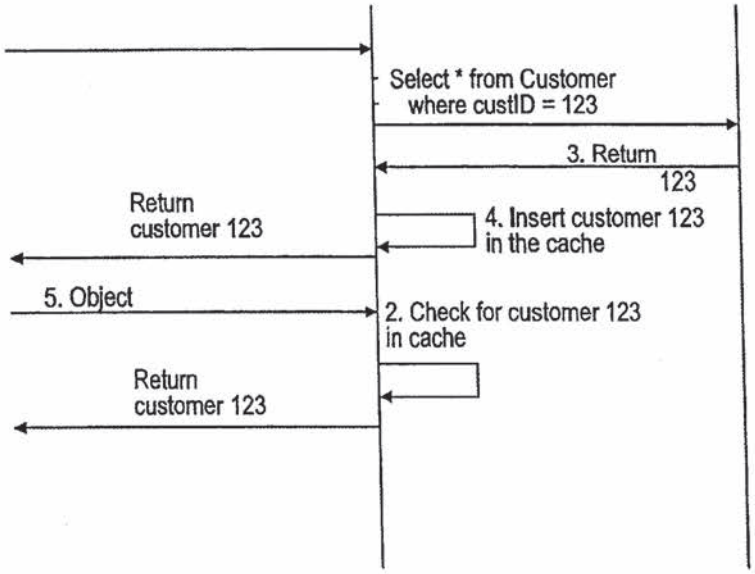


Fig. 172

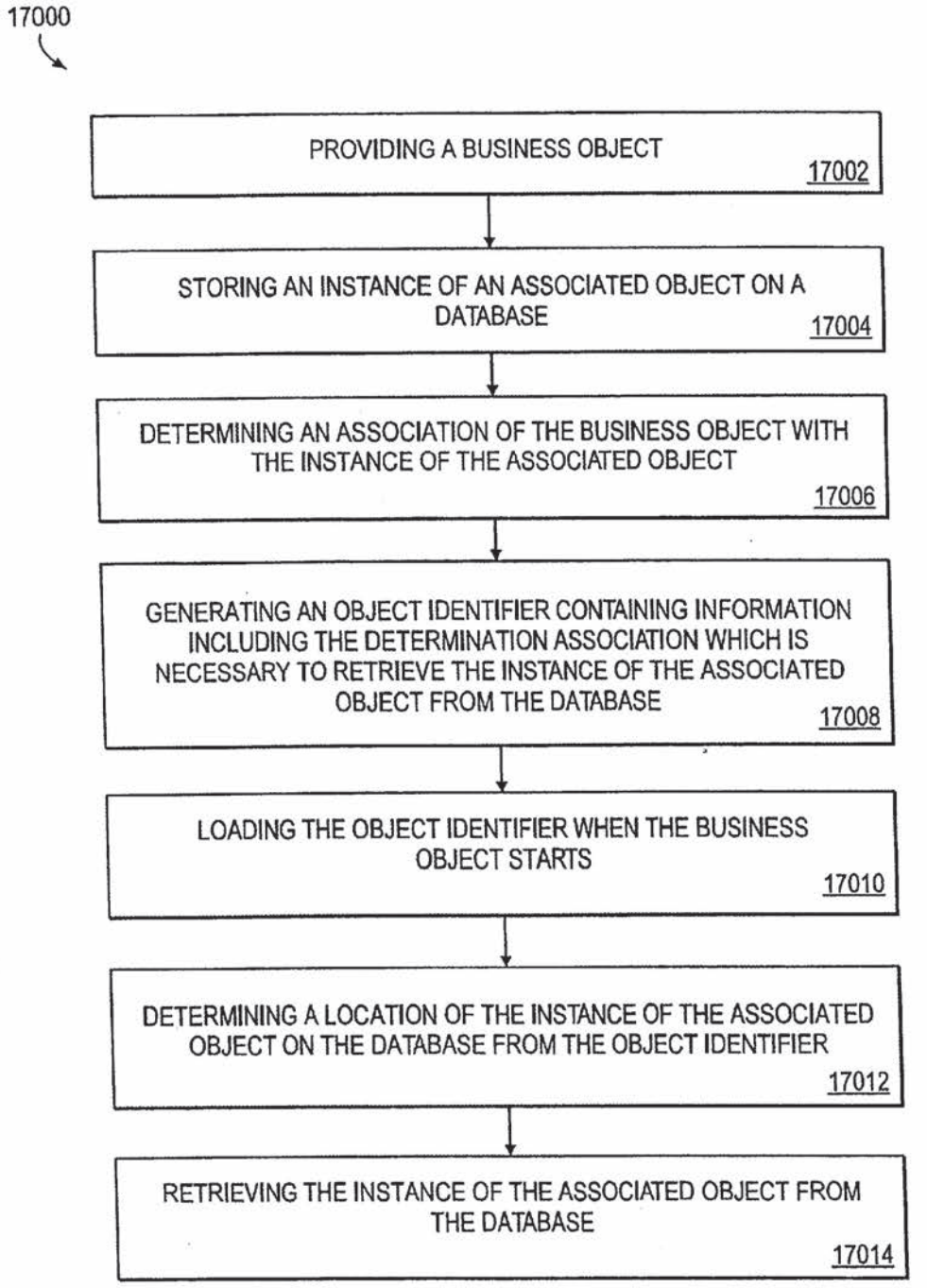

17000
PROVIDING A BUSINESS OBJECT
17002
STORING AN INSTANCE OF AN ASSOCIATED OBJECT ON A DATABASE
17004
DETERMINING AN ASSOCIATION OF THE BUSINESS OBJECT WITH THE INSTANCE OF THE ASSOCIATED OBJECT
17006
GENERATING AN OBJECT IDENTIFIER CONTAINING INFORMATION INCLUDING THE DETERMINATION ASSOCIATION WHICH IS NECESSARY TO RETRIEVE THE INSTANCE OF THE ASSOCIATED OBJECT FROM THE DATABASE
17008
LOADING THE OBJECT IDENTIFIER WHEN THE BUSINESS OBJECT STARTS
17010
DETERMINING A LOCATION OF THE INSTANCE OF THE ASSOCIATED OBJECT ON THE DATABASE FROM THE OBJECT IDENTIFIER
17012
RETRIEVING THE INSTANCE OF THE ASSOCIATED OBJECT FROM THE DATABASE
17014

Fig. 170

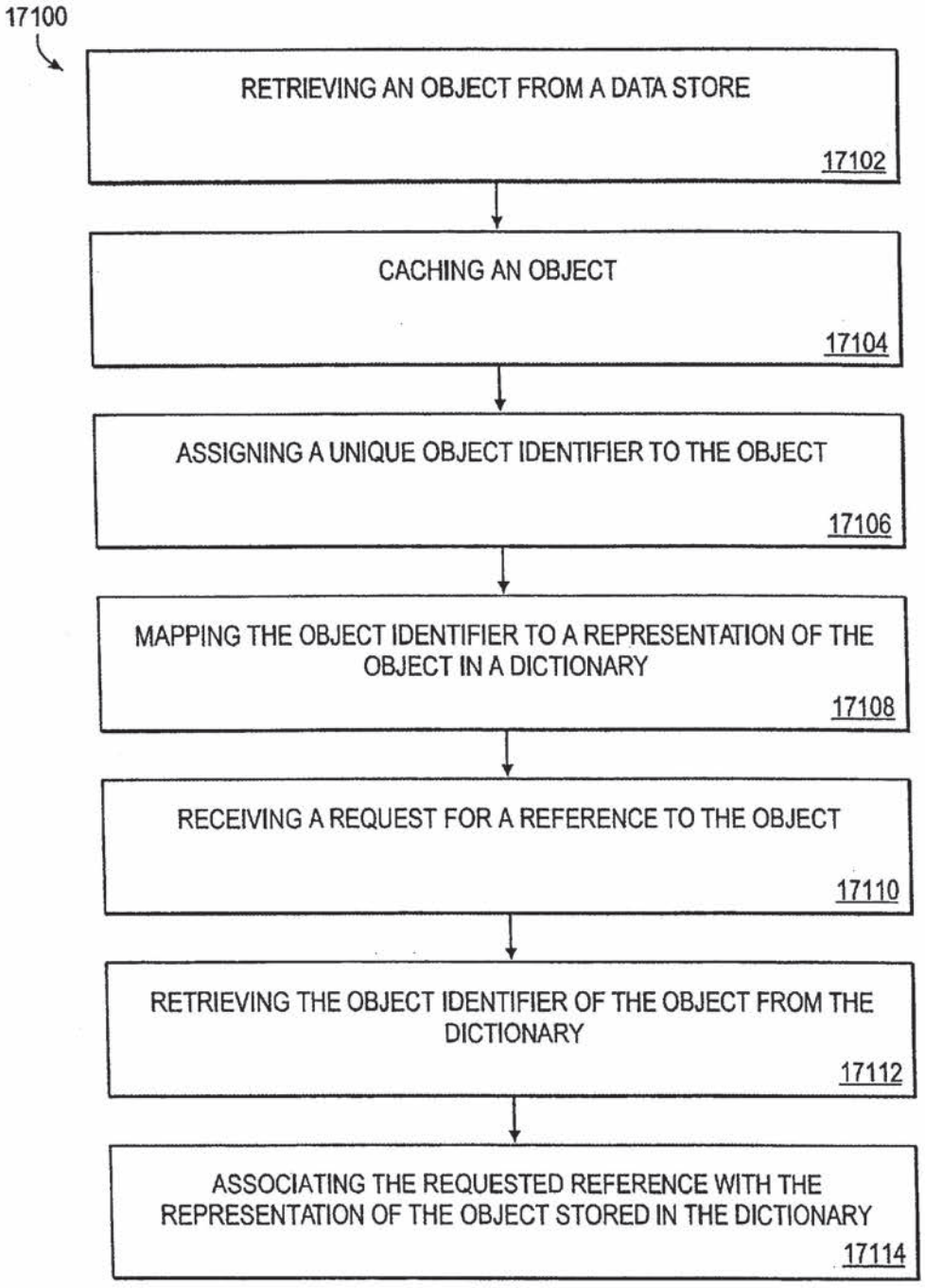

17100
RETRIEVING AN OBJECT FROM A DATA STORE
17102
CACHING AN OBJECT
17104
ASSIGNING A UNIQUE OBJECT IDENTIFIER TO THE OBJECT
17106
MAPPING THE OBJECT IDENTIFIER TO A REPRESENTATION OF THE OBJECT IN A DICTIONARY
17108
RECEIVING A REQUEST FOR A REFERENCE TO THE OBJECT
17110
RETRIEVING THE OBJECT IDENTIFIER OF THE OBJECT FROM THE DICTIONARY
17112
ASSOCIATING THE REQUESTED REFERENCE WITH THE REPRESENTATION OF THE OBJECT STORED IN THE DICTIONARY
17114

Fig. 171

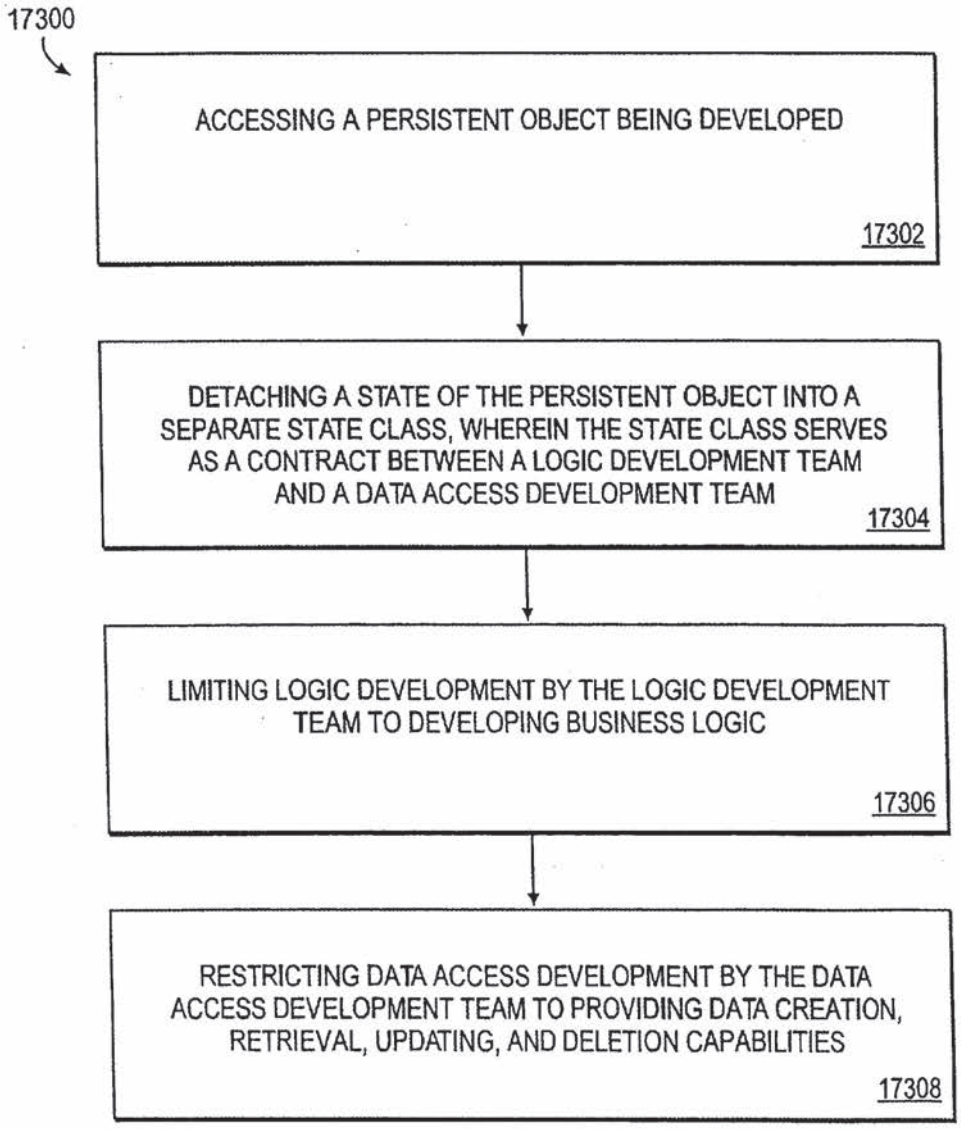

17300
ACCESSING A PERSISTENT OBJECT BEING DEVELOPED
17302
DETACHING A STATE OF THE PERSISTENT OBJECT INTO A SEPARATE STATE CLASS, WHEREIN THE STATE CLASS SERVES AS A CONTRACT BETWEEN A LOGIC DEVELOPMENT TEAM AND A DATA ACCESS DEVELOPMENT TEAM
17304
LIMITING LOGIC DEVELOPMENT BY THE LOGIC DEVELOPMENT TEAM TO DEVELOPING BUSINESS LOGIC
17306
RESTRICTING DATA ACCESS DEVELOPMENT BY THE DATA ACCESS DEVELOPMENT TEAM TO PROVIDING DATA CREATION, RETRIEVAL, UPDATING, AND DELETION CAPABILITIES
17308

Fig. 173

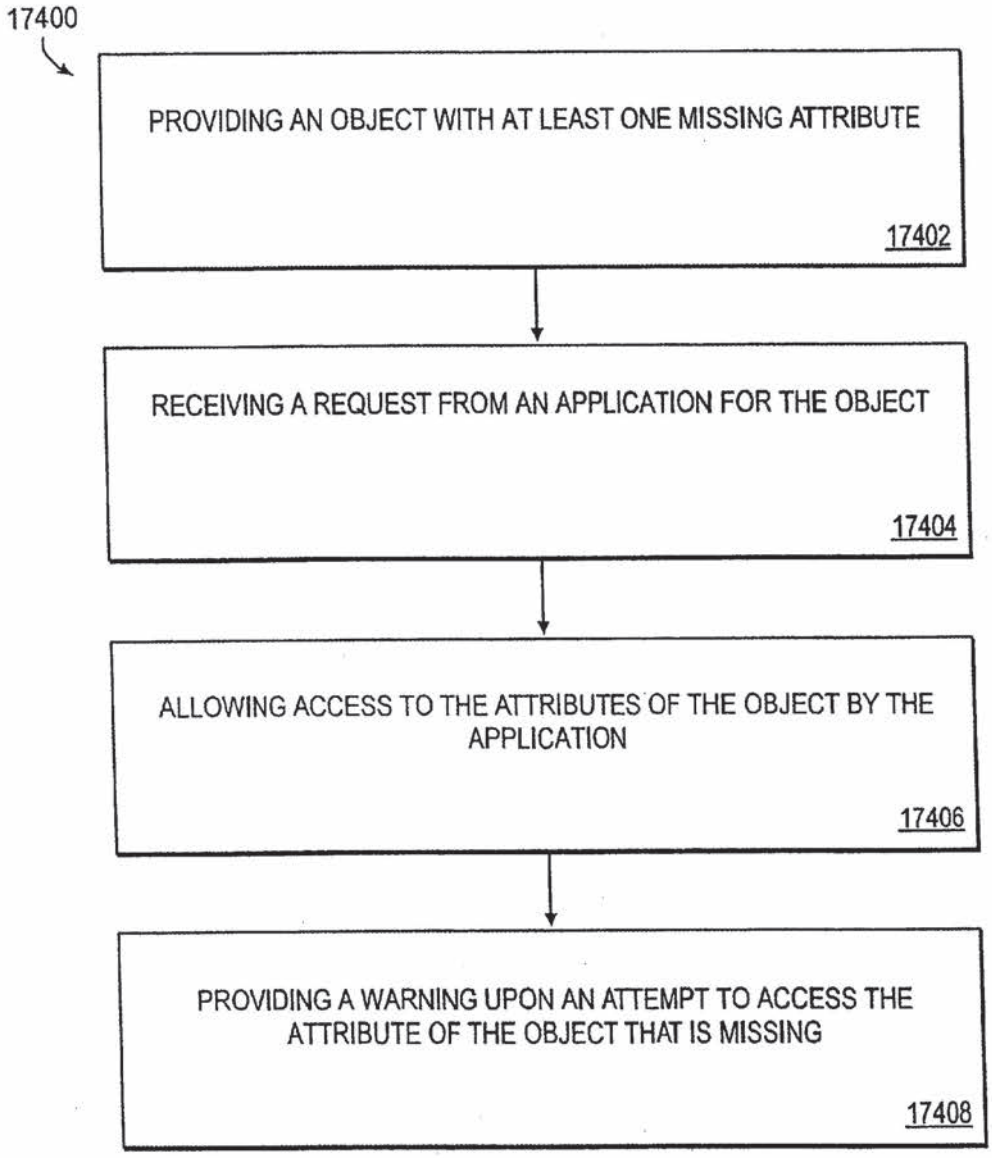

17400
PROVIDING AN OBJECT WITH AT LEAST ONE MISSING ATTRIBUTE
17402
RECEIVING A REQUEST FROM AN APPLICATION FOR THE OBJECT
17404
ALLOWING ACCESS TO THE ATTRIBUTES OF THE OBJECT BY THE APPLICATION
17406
PROVIDING A WARNING UPON AN ATTEMPT TO ACCESS THE ATTRIBUTE OF THE OBJECT THAT IS MISSING
17408

Fig. 174

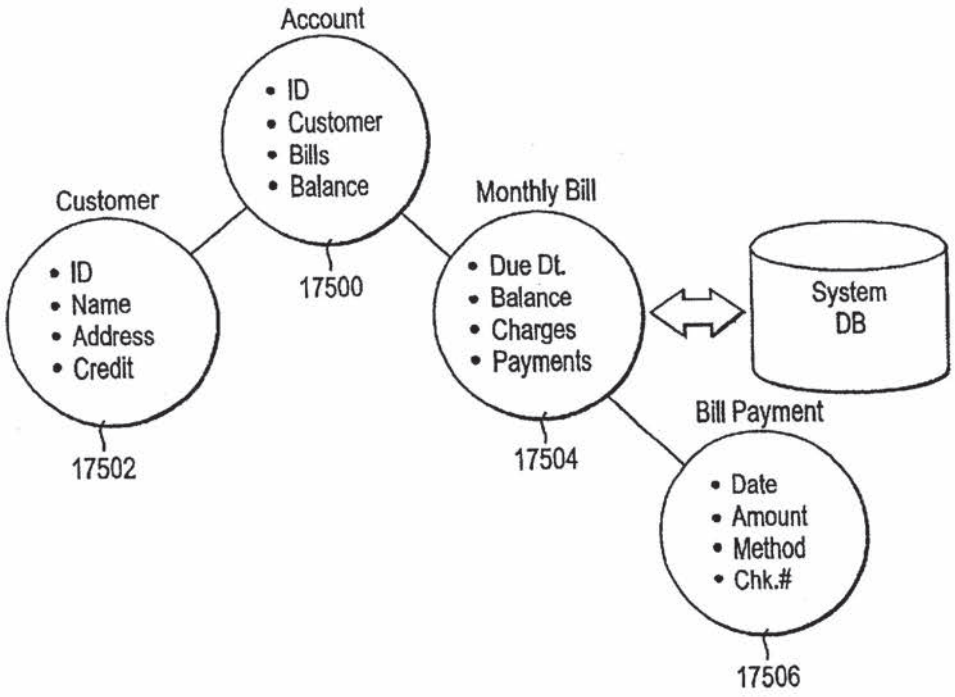


Fig. 175

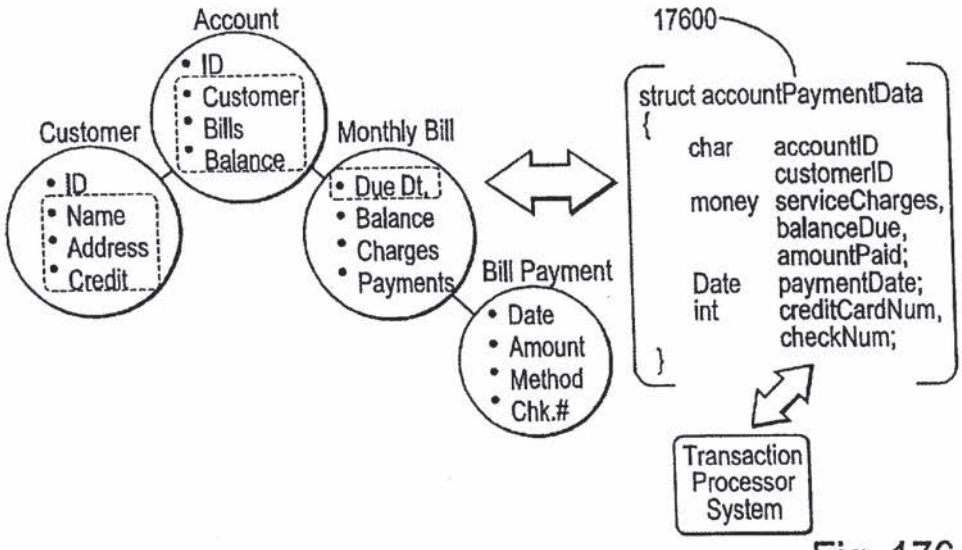


Fig. 176

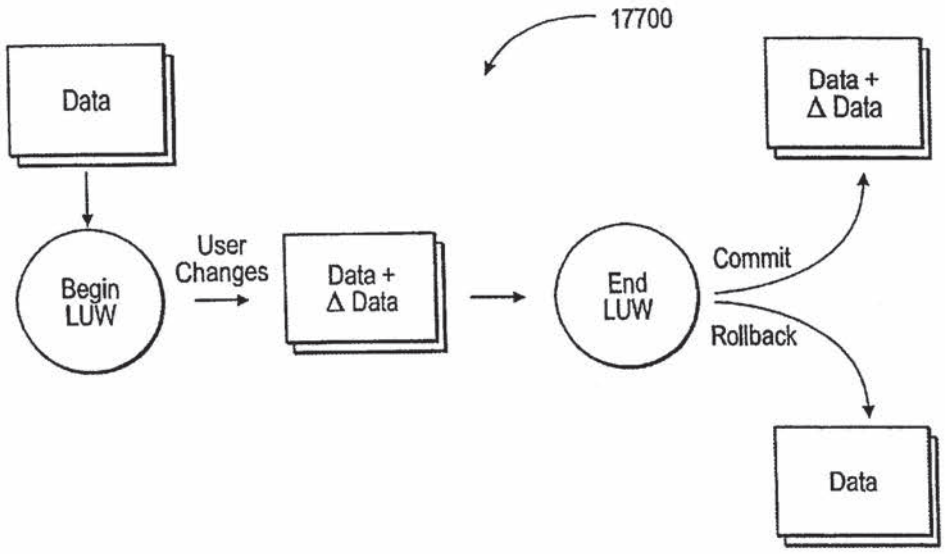


Fig. 177

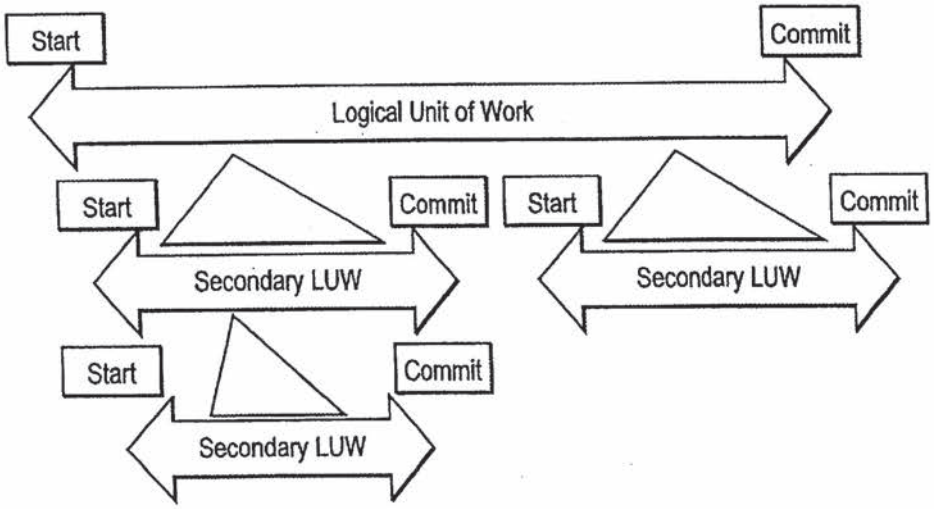


Fig. 178

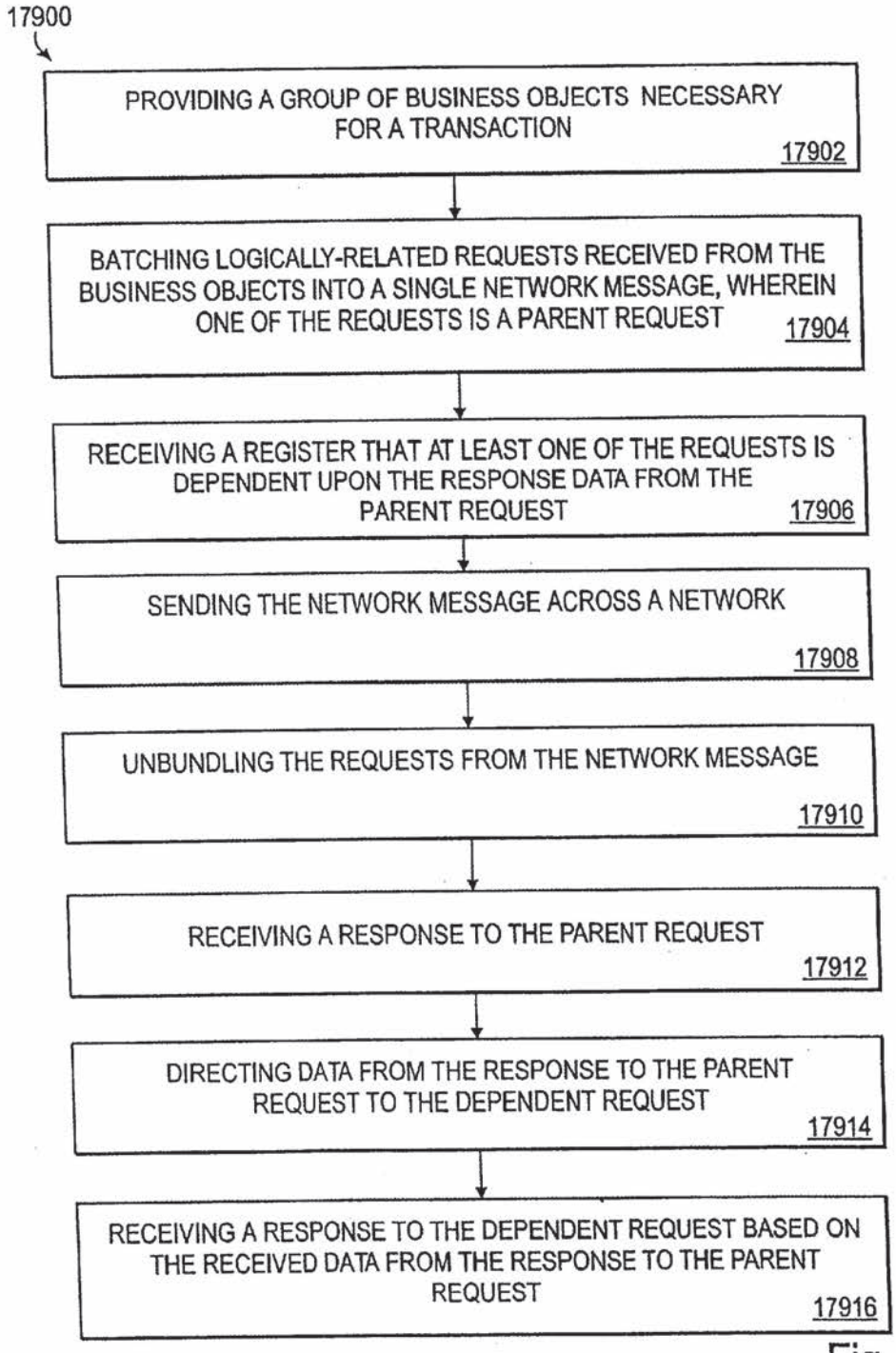

17900
PROVIDING A GROUP OF BUSINESS OBJECTS NECESSARY FOR A TRANSACTION
17902
BATCHING LOGICALLY-RELATED REQUESTS RECEIVED FROM THE BUSINESS OBJECTS INTO A SINGLE NETWORK MESSAGE, WHEREIN ONE OF THE REQUESTS IS A PARENT REQUEST
17904
RECEIVING A REGISTER THAT AT LEAST ONE OF THE REQUESTS IS DEPENDENT UPON THE RESPONSE DATA FROM THE PARENT REQUEST
17906
SENDING THE NETWORK MESSAGE ACROSS A NETWORK
17908
UNBUNDLING THE REQUESTS FROM THE NETWORK MESSAGE
17910
RECEIVING A RESPONSE TO THE PARENT REQUEST
17912
DIRECTING DATA FROM THE RESPONSE TO THE PARENT REQUEST TO THE DEPENDENT REQUEST
17914
RECEIVING A RESPONSE TO THE DEPENDENT REQUEST BASED ON THE RECEIVED DATA FROM THE RESPONSE TO THE PARENT REQUEST
17916

Fig. 179

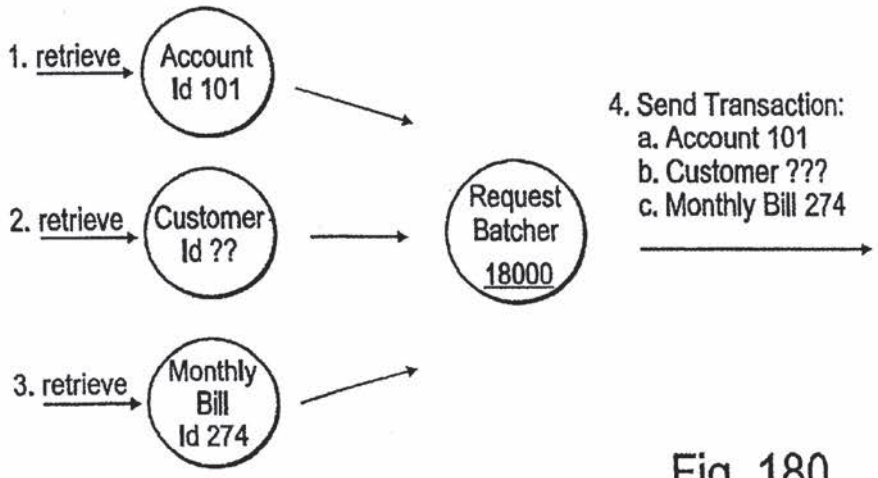


Fig. 180

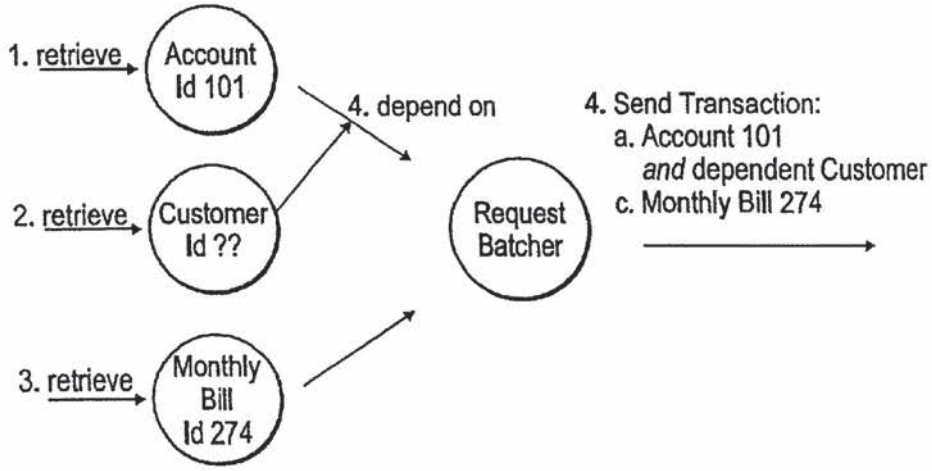


Fig. 181

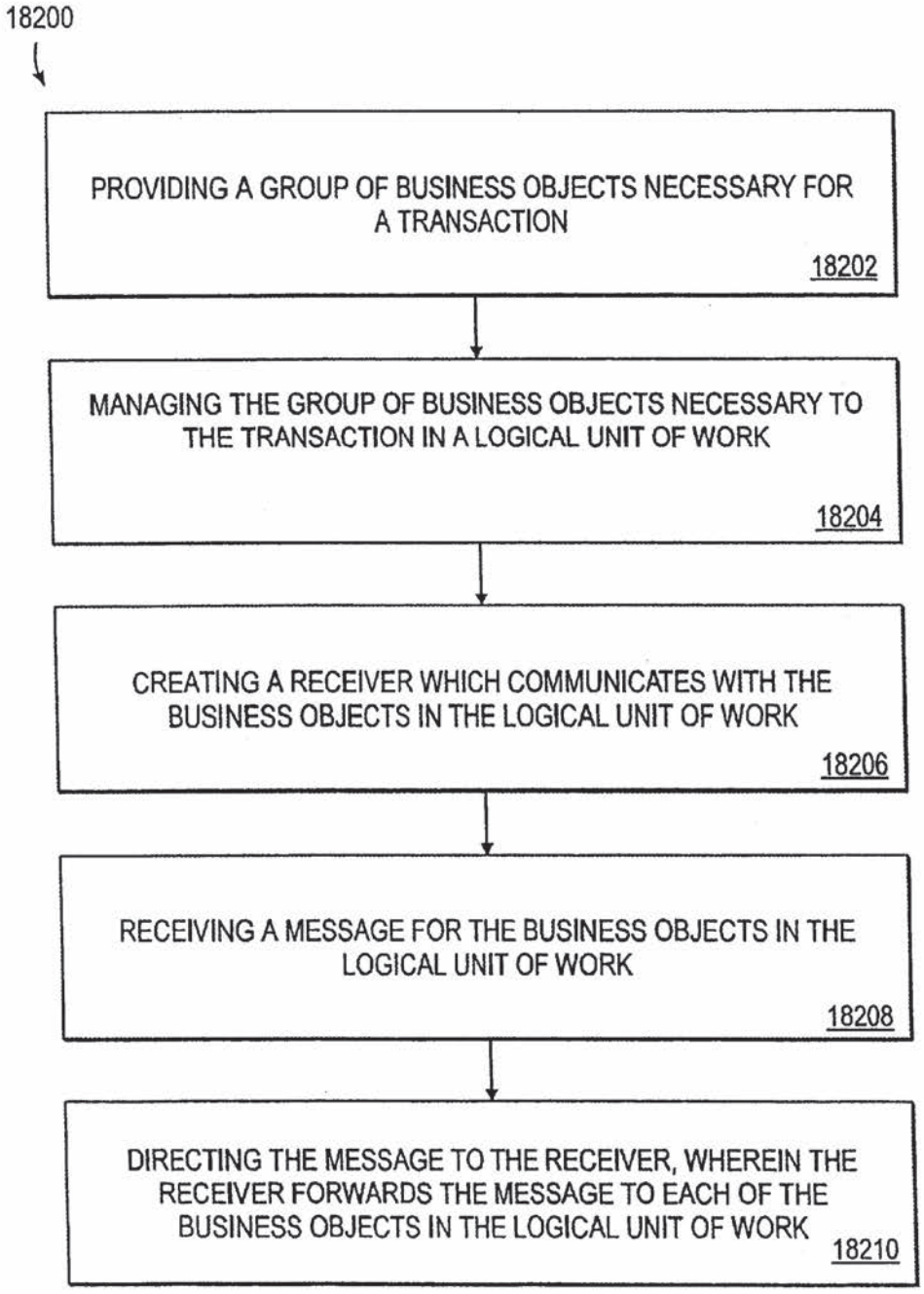

18200
PROVIDING A GROUP OF BUSINESS OBJECTS NECESSARY FOR A TRANSACTION
18202
MANAGING THE GROUP OF BUSINESS OBJECTS NECESSARY TO THE TRANSACTION IN A LOGICAL UNIT OF WORK
18204
CREATING A RECEIVER WHICH COMMUNICATES WITH THE BUSINESS OBJECTS IN THE LOGICAL UNIT OF WORK
18206
RECEIVING A MESSAGE FOR THE BUSINESS OBJECTS IN THE LOGICAL UNIT OF WORK
18208
DIRECTING THE MESSAGE TO THE RECEIVER, WHEREIN THE RECEIVER FORWARDS THE MESSAGE TO EACH OF THE BUSINESS OBJECTS IN THE LOGICAL UNIT OF WORK
18210

Fig. 182

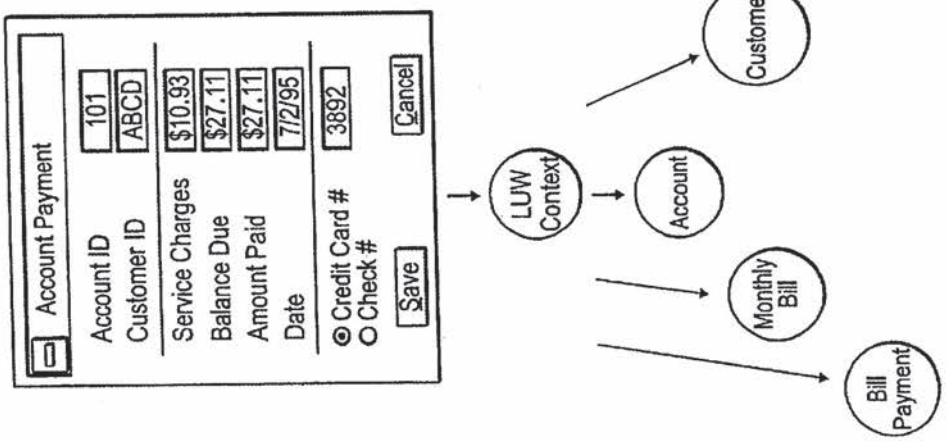


Fig. 184

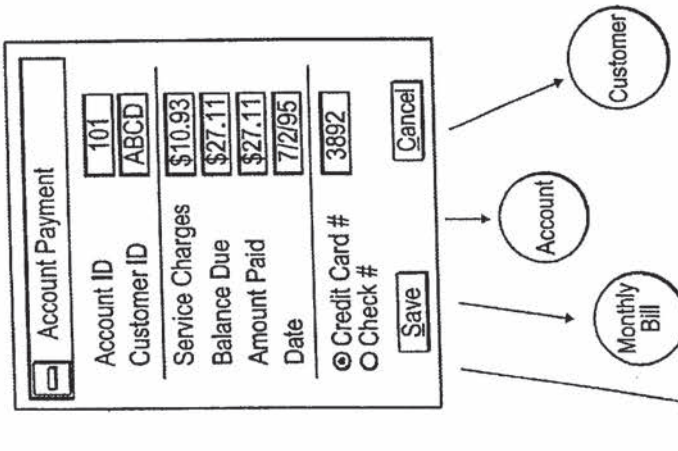


Fig. 183

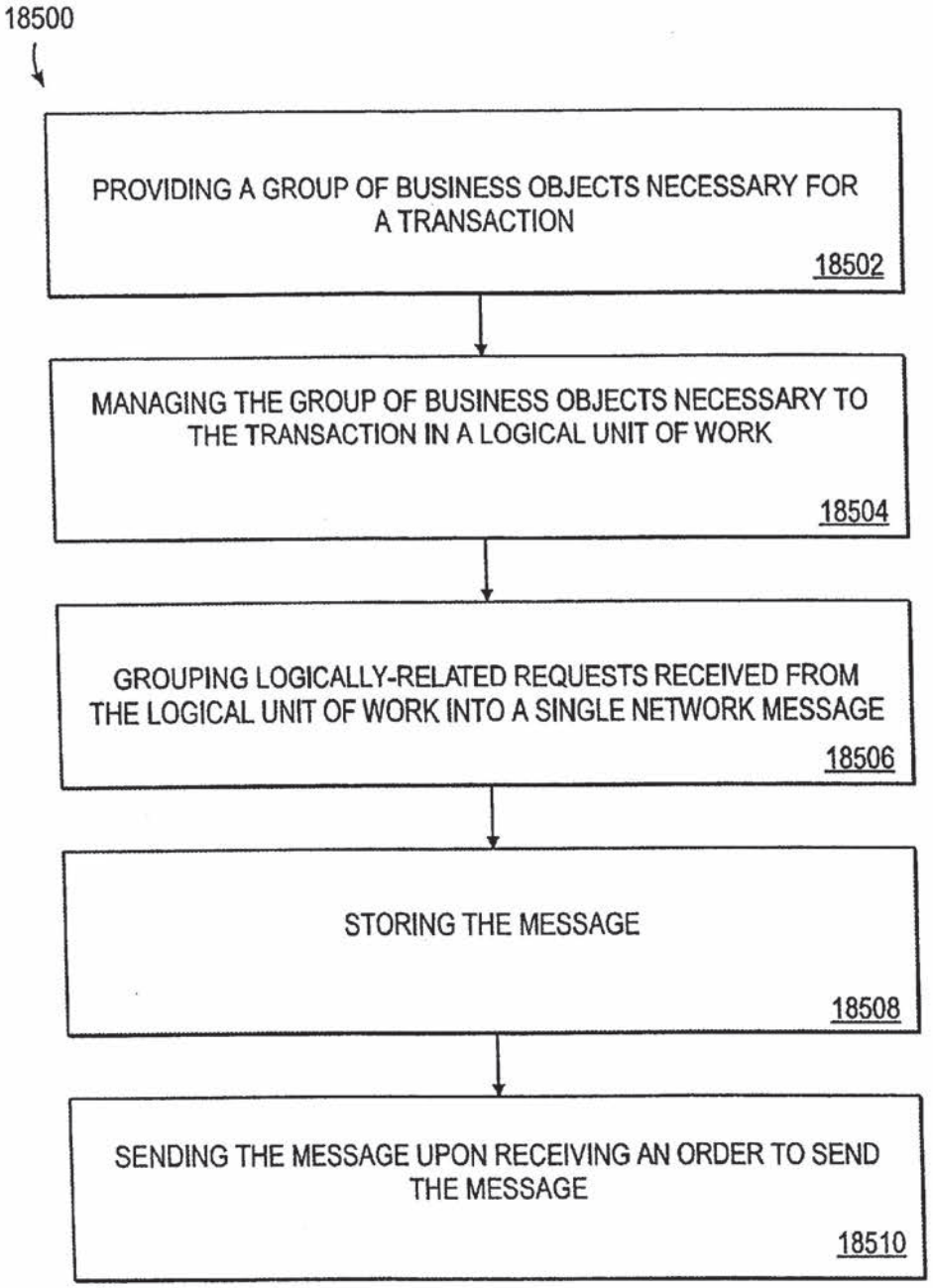

18500
PROVIDING A GROUP OF BUSINESS OBJECTS NECESSARY FOR A TRANSACTION
18502
MANAGING THE GROUP OF BUSINESS OBJECTS NECESSARY TO THE TRANSACTION IN A LOGICAL UNIT OF WORK
18504
GROUPING LOGICALLY-RELATED REQUESTS RECEIVED FROM THE LOGICAL UNIT OF WORK INTO A SINGLE NETWORK MESSAGE
18506
STORING THE MESSAGE
18508
SENDING THE MESSAGE UPON RECEIVING AN ORDER TO SEND THE MESSAGE
18510

Fig. 185

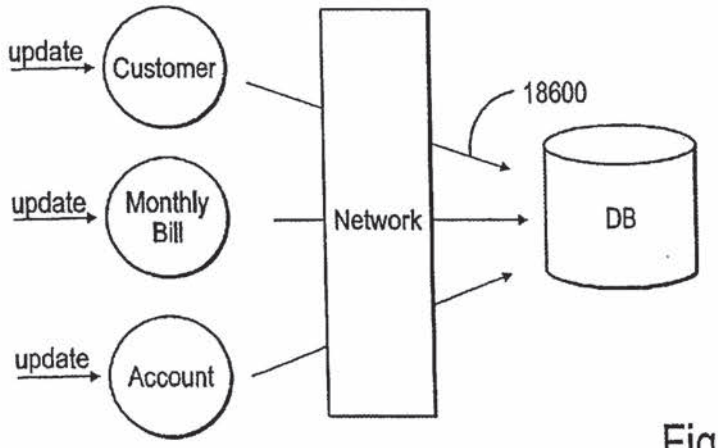


Fig. 186

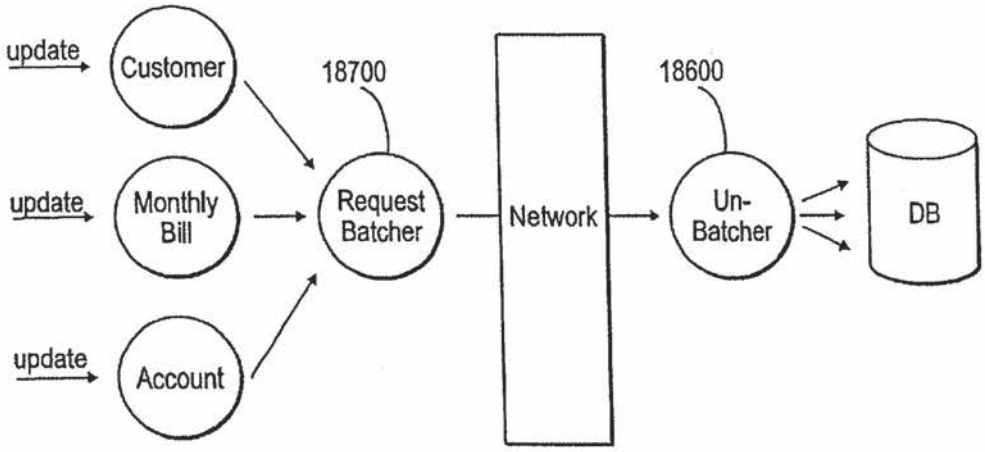


Fig. 187

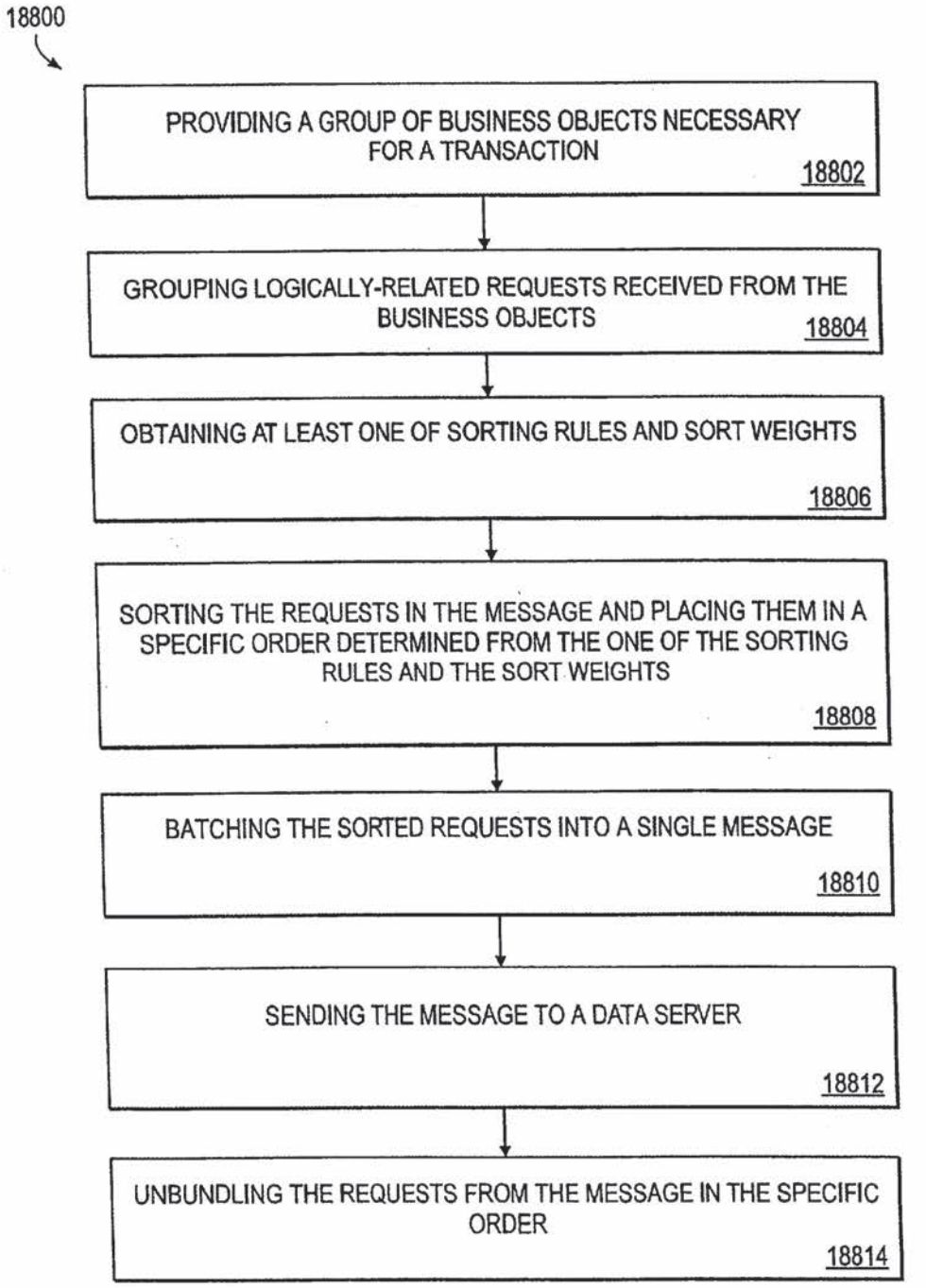

18800
PROVIDING A GROUP OF BUSINESS OBJECTS NECESSARY FOR A TRANSACTION
18802
GROUPING LOGICALLY-RELATED REQUESTS RECEIVED FROM THE BUSINESS OBJECTS
18804
OBTAINING AT LEAST ONE OF SORTING RULES AND SORT WEIGHTS
18806
SORTING THE REQUESTS IN THE MESSAGE AND PLACING THEM IN A SPECIFIC ORDER DETERMINED FROM THE ONE OF THE SORTING RULES AND THE SORT WEIGHTS
18808
BATCHING THE SORTED REQUESTS INTO A SINGLE MESSAGE
18810
SENDING THE MESSAGE TO A DATA SERVER
18812
UNBUNDLING THE REQUESTS FROM THE MESSAGE IN THE SPECIFIC ORDER
18814

Fig. 188

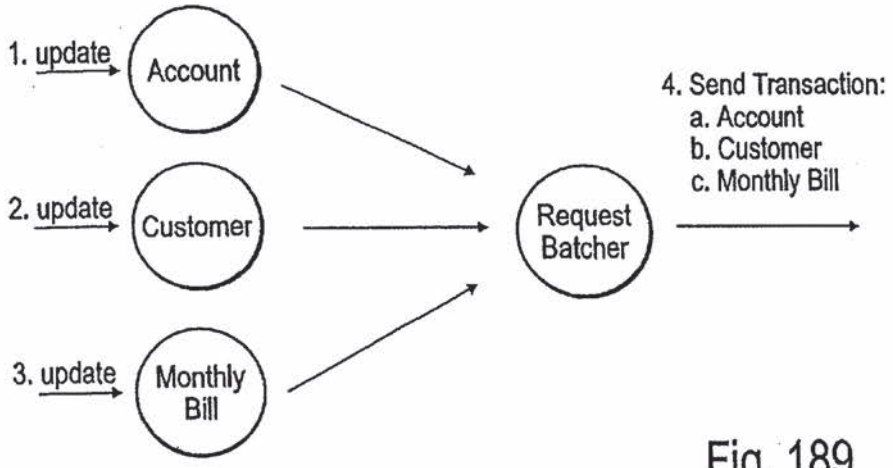


Fig. 189

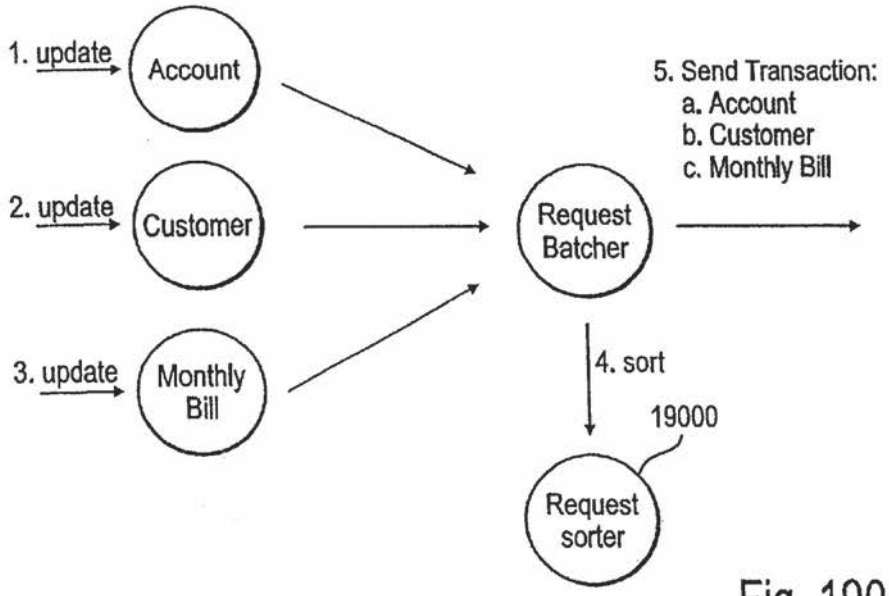


Fig. 190

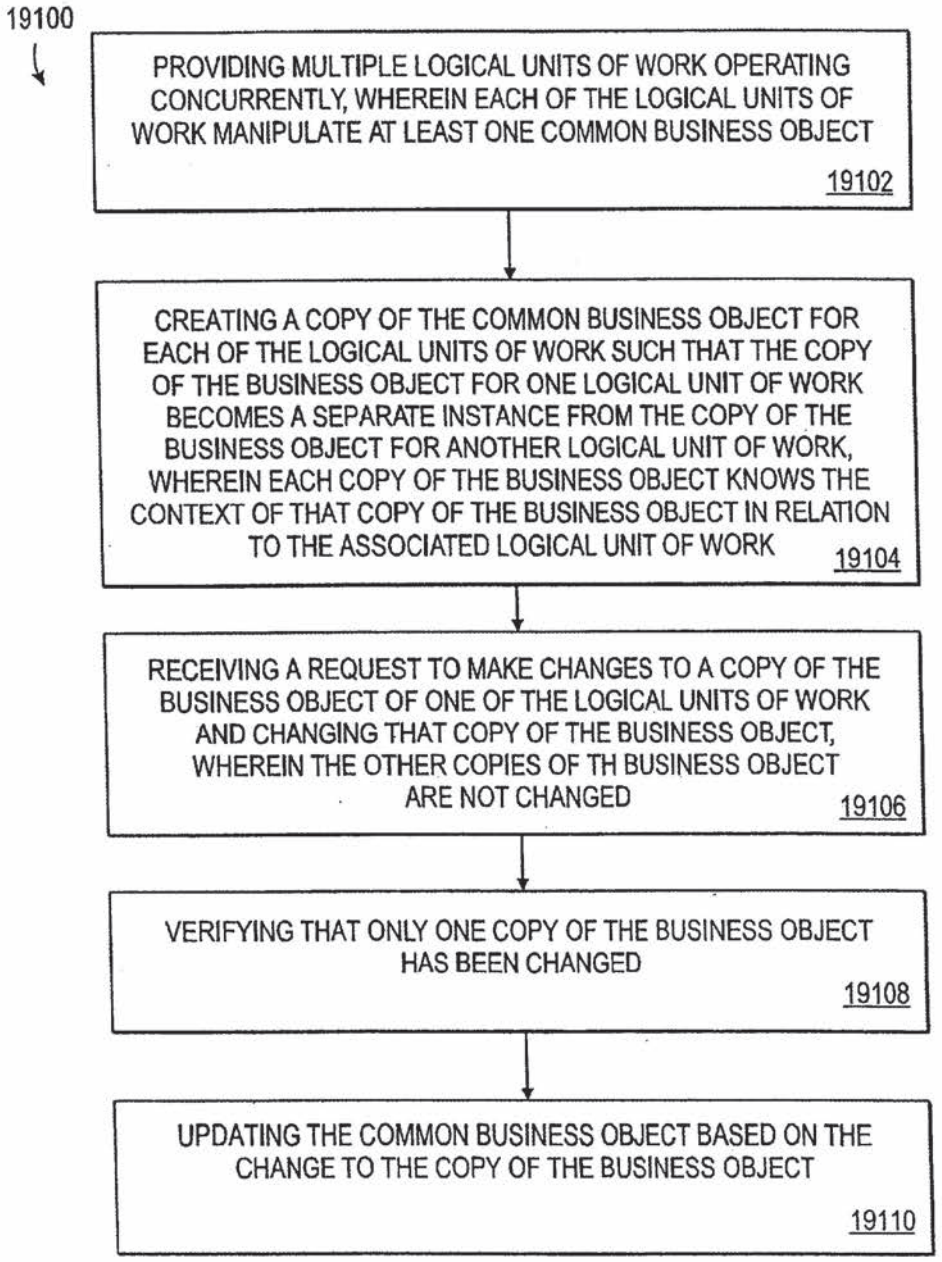

19100
PROVIDING MULTIPLE LOGICAL UNITS OF WORK OPERATING CONCURRENTLY, WHEREIN EACH OF THE LOGICAL UNITS OF WORK MANIPULATE AT LEAST ONE COMMON BUSINESS OBJECT
19102
CREATING A COPY OF THE COMMON BUSINESS OBJECT FOR EACH OF THE LOGICAL UNITS OF WORK SUCH THAT THE COPY OF THE BUSINESS OBJECT FOR ONE LOGICAL UNIT OF WORK BECOMES A SEPARATE INSTANCE FROM THE COPY OF THE BUSINESS OBJECT FOR ANOTHER LOGICAL UNIT OF WORK, WHEREIN EACH COPY OF THE BUSINESS OBJECT KNOWS THE CONTEXT OF THAT COPY OF THE BUSINESS OBJECT IN RELATION TO THE ASSOCIATED LOGICAL UNIT OF WORK
19104
RECEIVING A REQUEST TO MAKE CHANGES TO A COPY OF THE BUSINESS OBJECT OF ONE OF THE LOGICAL UNITS OF WORK AND CHANGING THAT COPY OF THE BUSINESS OBJECT, WHEREIN THE OTHER COPIES OF TH BUSINESS OBJECT ARE NOT CHANGED
19106
VERIFYING THAT ONLY ONE COPY OF THE BUSINESS OBJECT HAS BEEN CHANGED
19108
UPDATING THE COMMON BUSINESS OBJECT BASED ON THE CHANGE TO THE COPY OF THE BUSINESS OBJECT
19110

Fig. 191

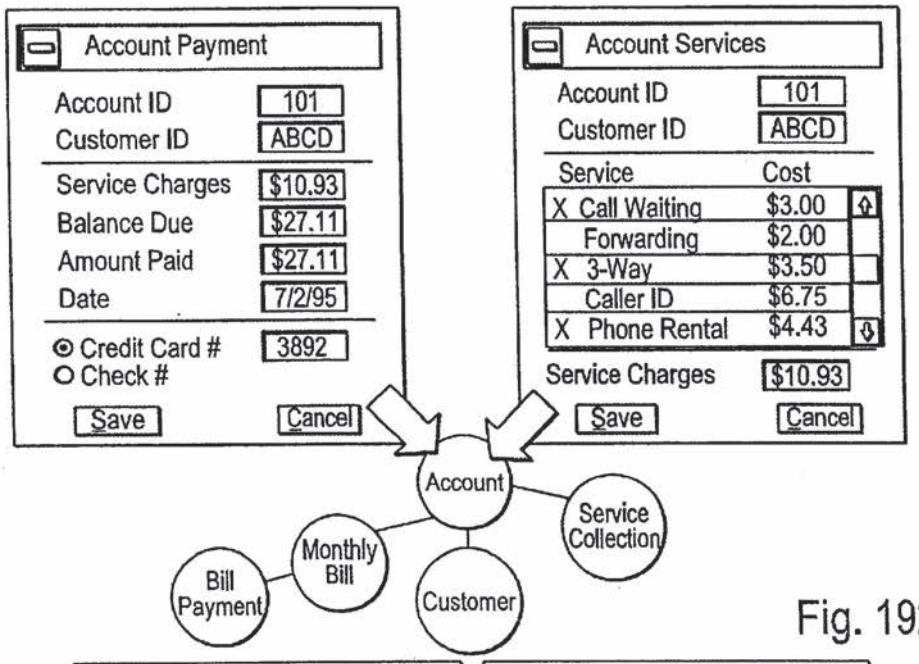


Fig. 192

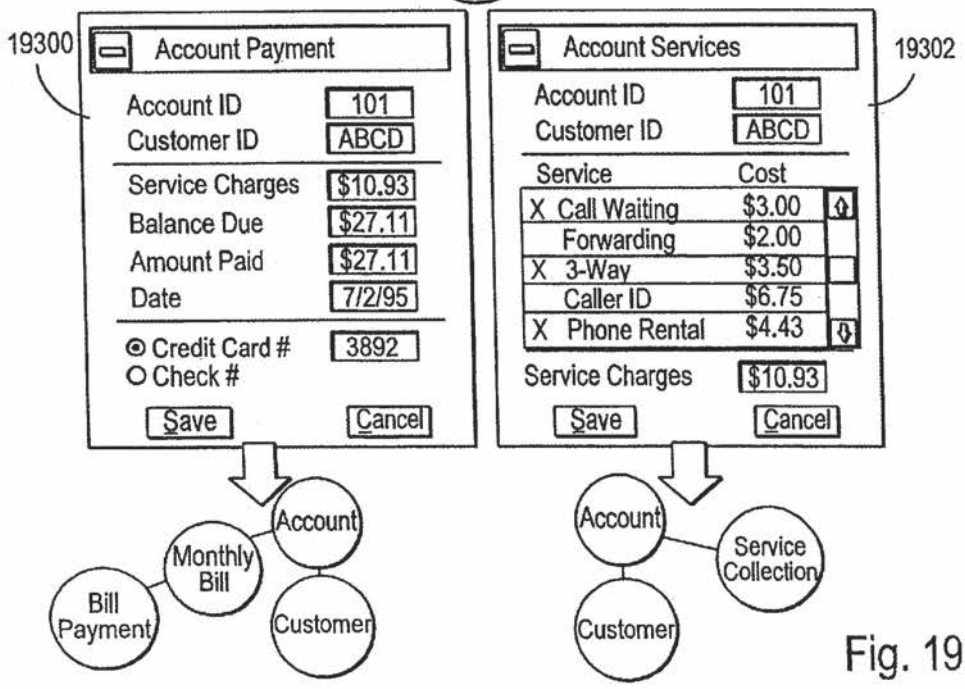


Fig. 193

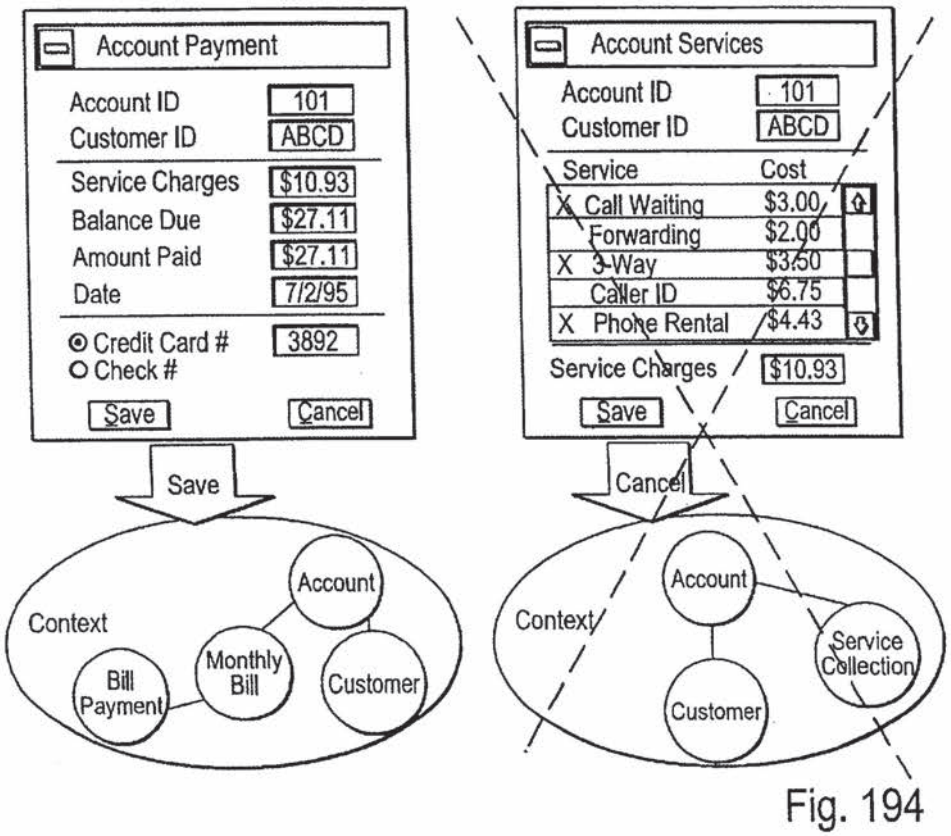


Fig. 194

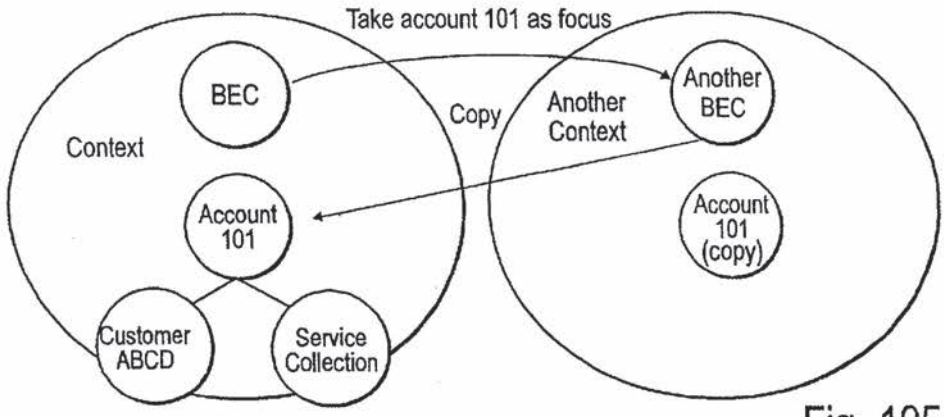


Fig. 195

EFFICIENT SERVER SIDE DATA RETRIEVAL FOR EXECUTION OF CLIENT SIDE APPLICATIONS

CROSS REFERENCE TO RELATED APPLICATIONS

This application is related to United States Patent Applications entitled A SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR A DEVELOPMENT ARCHITECTURE FRAMEWORK U.S. patent application Ser. No. 09/387,747, filed Aug. 31, 1999 and A SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR MAINTENANCE AND ADMINISTRATION IN AN E-COMMERCE APPLICATION FRAMEWORK U.S. patent application Ser. No. 09/387,318, both of which are filed concurrently herewith and which are incorporated by reference in their entirety.

FIELD OF THE INVENTION

The present invention relates to software patterns and more particularly to updating a server while minimizing network traffic and maintaining an acceptable level of system performance.

BACKGROUND OF THE INVENTION

An important use of computers is the transfer of information over a network Currently, the largest computer network in existence is the Internet. The Internet is a worldwide interconnection of computer networks that communicate using a common protocol. Millions of computers, from low end personal computers to high-end super computers are coupled to the Internet.

The Internet grew out of work funded in the 1960s by the U.S. Defense Department's Advanced Research Projects Agency. For a long time, Internet was used by researchers in universities and national laboratories to share information. As the existence of the Internet became more widely known, many users outside of the academic/research community (e.g., employees of large corporations) started to use Internet to carry electronic mail.

In 1989, a new type of information system known as the World-Wide-Web ("the Web") was introduced to the Internet. Early development of the Web took place at CERN, the European Particle Physics Laboratory. The Web is a wide-area hypermedia information retrieval system aimed to give wide access to a large universe of documents. At that time, the Web was known to and used by the academic/research community only. There was no easily available tool which allows a technically untrained person to access the Web.

In 1993, researchers at the National Center for Supercomputing Applications (NCSA) released a Web browser called "Mosaic" that implemented a graphical user interface (GUI). Mosaic's graphical user interface was simple to learn yet powerful. The Mosaic browser allows a user to retrieve documents from the World-Wide-Web using simple point-and-click commands. Because the user does not have to be technically trained and the browser is pleasant to use, it has the potential of opening up the Internet to the masses.

The architecture of the Web follows a conventional client-server model. The terms "client" and "server" are used to refer to a computer's general role as a requester of data (the client) or provider of data (the server). Under the Web environment, Web browsers reside in clients and Web documents reside in servers. Web clients and Web servers communicate using a protocol called "HyperText Transfer Protocol" (HTTP). A browser opens a connection to a server and initiates a request for a document. The server delivers the requested document, typically in the form of a text document coded in a standard Hypertext Markup Language (HTML) format, and when the connection is closed in the above interaction, the server serves a passive role, i.e., it accepts commands from the client and cannot request the client to perform any action.

The communication model under the conventional Web environment provides a very limited level of interaction between clients and servers. In many systems, increasing the level of interaction between components in the systems often makes the systems more robust, but increasing the interaction increases the complexity of the interaction and typically slows the rate of the interaction. Thus, the conventional Web environment provides less complex, faster interactions because of the Web's level of interaction between clients and servers.

SUMMARY OF THE INVENTION

A system, method, and article of manufacture are provided for efficiently retrieving data. A total amount of data required for an application executed by a client is determined. In a single call, the total amount of data from a server is requested over a network. All of the data is bundled into a data structure by the server in response to the single call. The bundled data structure is sent to the client over the network and the data of the data structure is cached on the client. The cached data of the data structure is used as needed during execution of the application on the client.

In an aspect of the present invention, the data structure may be bundled on the server by a business object. In an additional aspect of the present invention, the business object may be instantiated by an action of the client. In another aspect of the present invention, the network may be at least one of a local area network and a wide area network.

In a further aspect of the present invention, the request may be administered by a proxy component. In even another aspect of the present invention, the data structure may contain no logic.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be better understood when consideration is given to the following detailed description thereof. Such description makes reference to the annexed drawings wherein:

FIG. **1** is a schematic diagram of a hardware implementation of one embodiment of the present invention;

FIG. **2** is a flow diagram illustrating a high level overview of an architecture;

FIG. **3** shows the dependencies of three architecture frameworks;

FIG. **4** illustrates a delivery vehicle matrix;

FIG. **5** illustrates a Delivery Vehicle Cube;

FIG. **6** is a flow diagram depicting considerations to be taken into consideration when identifying the core technologies to be used in an architecture;

FIG. **7** is a chart that can be utilized to determine whether to use Netcentric technology;

FIG. **8** is a chart that can be utilized to determine whether to use Client Server technology;

FIG. **9** is a chart that can be utilized to determine whether to use Host technology;

FIG. **10** illustrates the services of a Netcentric Architecture Framework in accordance with one embodiment of the present invention;

FIG. **11** is a detailed diagram of some of the components of the Netcentric Architecture Framework found in FIG. **10**;

FIG. **12** is a detailed diagram of other components of the Netcentric Architecture Framework found in FIG. **10**;

FIG. **13** illustrates several components of the Presentation area of the Netcentric Architecture Framework;

FIG. **14** illustrates several components of the Information Services of the present invention;

FIG. **15** depicts the four major categories of functionality that the Network services provided by the Communications Services are grouped into;

FIG. **16** illustrates File Sharing services;

FIG. **17** depicts Message Passing services;

FIG. **18** depicts Message Queuing services;

FIG. **19** illustrates Publish and Subscribe services;

FIG. **20** depicts Streaming, in which a real-time data stream is transferred;

FIG. **21** illustrates CORBA-based Object Messaging;

FIG. **22** illustrates COM Messaging;

FIG. **23** represents CTI Messaging;

FIG. **24** illustrates various components of the Communication Fabric of the present invention;

FIG. **25** illustrates the two categories of the Physical Media;

FIG. **26** illustrates several of the components of the Transaction areas of the Netcentric Architecture Framework;

FIG. **27** illustrates various components of the Environmental Services of the Netcentric Architecture Framework;

FIG. **28** illustrates the Base Services of the Netcentric Architecture Framework;

FIG. **29** shows the major components of the reporting application framework;

FIG. **30** illustrates an example of how a custom report architecture relates to a workstation platform technology architecture;

FIG. **31** describes the relationships between the major components of the report process and the report writer process;

FIG. **32** shows the module hierarchy for the custom report process;

FIG. **33** depicts the various components of the Business Logic portion of the Netcentric Architecture Framework;

FIG. **34** illustrates a relationship between major themes that impact aspects of software development and management;

FIG. **35** illustrates how components are viewed from different perspectives;

FIG. **36** shows a relationship between business components and partitioned business components;

FIG. **37** shows how a Billing Business Component may create an invoice;

FIG. **38** illustrates the relationship between the spectrum of Business Components and the types of Partitioned Business Components;

FIG. **39** illustrates the flow of workflow, dialog flow, and/or user interface designs to a User Interface Component;

FIG. **40** is a diagram of an Application Model which illustrates how the different types of Partitioned Business Components might interact with each other;

FIG. **41** illustrates what makes up a Partitioned Business Component;

FIG. **42** illustrates the role of patterns and frameworks;

FIG. **43** illustrates this Business Component Identifying Methodology including both Planning and Delivering stages;

FIG. **44** shows a high level picture of application component interaction for an Order Entry system;

FIG. **45** illustrates a traditional organization structure including an activities component, a credit/collections component, a billing component, and a finance component;

FIG. **46** provides an illustration of a horizontal organization model;

FIG. **47** illustrates a workcell organization approach including an activities component, a credit/collections component, a billing component, and a finance component;

FIG. **48** illustrates the Enterprise Information Architecture (EIA) model;

FIG. **49** illustrates a V-model of Verification, Validation, and Testing;

FIG. **50** portrays of a development architecture with a seamless integration of tools which can be plugged in for the capture and communication of particular deliverables;

FIG. **51** shows a design architecture with the compromises made for today's component construction environment;

FIG. **52** illustrates a business process to object mapping;

FIG. **53** is a diagram which illustrates a graph of resilience to change;

FIG. **54** illustrates a flowchart for a method for providing an abstraction factory pattern in accordance with an embodiment of the present invention;

FIG. **55** illustrates a flowchart for a method for representing a plurality of batch jobs of a system each with a unique class in accordance with an embodiment of the present invention;

FIG. **56** illustrates a class diagram of the batch job hierarchy;

FIG. **57** illustrates an object interaction graph of a possible implementation of the class diagram of FIG. **56**;

FIG. **58** illustrates a flowchart for a method for controlling access to data of a business object via an attribute dictionary in accordance with an embodiment of the present invention;

FIG. **59** illustrates a flowchart for a method for structuring batch activities for simplified reconfiguration in accordance with an embodiment of the present invention;

FIG. **60** illustrates the manner in which the AttributeDictionaryClient is the facade which delegates to the AttributeDictionary;

FIG. **61** depicts the use of the containsKey( ) method on the HashMap to ensure that the value will exist before the get( ) method is used;

FIG. **62** illustrates a method that dictates that any nullPointerException that is thrown would be caught and rethrown as the more user-friendly exception in the attribute dictionary pattern environment;

FIG. **63** illustrates the Get the Attribute Names method in the attribute dictionary pattern environment;

FIG. **64** illustrates a flowchart for a method for managing constants in a computer program in accordance with an embodiment of the present invention;

FIG. **65** illustrates a flowchart for a method for providing a fixed format stream-based communication system in accordance with an embodiment of the present invention;

FIG. **66** illustrates two systems communicating via a stream-based communication and using a common generic format to relay the meta-data information;

FIG. **67** illustrates an example of a Fixed Format message associated with the fixed format stream patterns;

FIG. **68** depicts the complete Fixed Format Stream pattern associated with the s fixed format stream patterns;

FIG. **69** illustrates fixed format contracts containing meta-data information for translating structured data onto and off of a stream;

FIG. **70** illustrates a Customer object in an object-based system streaming itself into a stream, the stream being sent to a non-object system, this stream being read and the data inserted into a relational database;

FIG. **71** illustrates a flowchart for a method for delivering service via a globally addressable interface in accordance with an embodiment of the present invention;

FIG. **72** depicts a client that is unable to find the services provided by a server via a network;

FIG. **73** illustrates the grouping of services using interfaces;

FIG. **74** illustrates a customer server publicly announcing its interfaces;

FIG. **75** illustrates a method including the registering and then locating of a globally addressable interface;

FIG. **76** illustrates the present invention using a method wherein a globally addressable interface is used to obtain data from a server;

FIG. **77** illustrates a flowchart for a method for affording access to a legacy system in accordance to an embodiment of the present invention;

FIG. **78** depicts the communication difficulties associated with Legacy Systems attempting to communicate with a client via a component integration architecture;

FIG. **79** illustrates homogenous interfaces from components which rectify the problems with Legacy Systems attempting to communicate with a client via a component integration architecture;

FIG. **80** shows how a Legacy Component is integrated into a component-based model;

FIG. **81** illustrates Legacy Wrapper Components of a Pure Legacy Wrapper Component including a Legacy Wrapper Component, a Component Adapter, a Legacy Integration Architecture, a Legacy Adapter, and a Legacy System;

FIG. **82** illustrates a Hybrid Component type of Legacy Wrapper Component;

FIG. **83** shows an abstract example of the control flow in a Legacy Component;

FIG. **84** illustrates a flowchart for a method for for delivering service via a locally addressable interface in accordance with an embodiment of the present invention;

FIG. **85** illustrates Problems with Globally Addressable Interfaces in a system including clients and servers with a plurality of interfaces;

FIG. **86** illustrates the manner in which the present invention uses a Locally Addressable Interface to hide functionality and lessen the load on the Naming or Trading Service;

FIG. **87** illustrates the manner in which the present invention obtains a Locally Addressable Interface;

FIG. **88** illustrates the method in which the present invention registers and then locates a Globally Addressable Interface;

FIG. **89** illustrates the manner in which the present invention uses a Globally Addressable Interface to obtain a Locally Addressable Interface to a specific Customer Object;

FIG. **90** illustrates a flowchart for a method for communicating a null value in accordance with an embodiment of the present invention;

FIG. **91** illustrates the problem associated with sending a NULL across many types of middleware;

FIG. **92** illustrates the manner in which the present invention passes a "null" structure across the middleware;

FIG. **93** depicts conversations with a "null" data structure;

FIG. **94** depicts conversations with a non-"null" data structure;

FIG. **95** illustrates a flowchart for a method for transmitting data from a server to a client via pages in accordance with an embodiment of the present invention;

FIG. **96** depicts the response time for a User Interface to display a list of customers in a list box;

FIG. **97** shows a request that returns a large amount of data;

FIG. **98** shows a graphical depiction of a paging communication pattern;

FIG. **99** illustrates a message trace diagram showing the interactions between a Client and a Server using Paging Communication to satisfy the previously mentioned scenario,

FIG. **100** illustrates a flowchart for a method for interfacing a naming service and a client with the naming service allowing access to a plurality of different sets of services from a plurality of globally addressable interfaces in accordance with an embodiment of the present invention;

FIG. **101** illustrates repeated requests to the Trader Service for the same interfaces;

FIG. **102** illustrates how a pool can be created that reuses GAI proxies;

FIG. **103** illustrates the implementation of a Refreshable Proxy Pool;

FIG. **104** illustrates the class relationships between the patterns primary classes;

FIG. **105** illustrates a flowchart for a method for providing a self-describing stream-based communication system in accordance with an embodiment of the present invention;

FIG. **106** illustrates two systems communicating via Stream-Based Communication and using a shared generic format to relay the meta-data information;

FIG. **107** illustrates an object-based system with a frequently changing object model communicating via Stream-Based Communication;

FIG. **108** illustrates a stream-based message that contains both message data and descriptive meta-data;

FIG. **109** illustrates the manner in which a message language defines how to parameterize the meta-data and put it on the stream;

FIG. **110** illustrates a Customer object in an object-based system streaming itself into a stream, the stream being sent to a non-object system, this stream being read and the data inserted into a relational database;

FIG. **111** illustrates a flowchart for a method for providing a stream-based communication system in accordance with an embodiment of the present invention;

FIG. **112** illustrates how systems of the present invention communicate over a communication mechanism that cannot inherently convey meta-data information;

FIG. **113** is an illustration of an object-based system communicating with a non-object system using a communication mechanism that cannot convey meta-data information;

FIG. **114** depicts an example of Stream Based Communication with two disparate systems communicating via stream-based communication;

FIG. **115** is an illustration of a Customer object in an object-based system streaming itself into a stream, the stream being sent to a non-object system, this stream being read and the information is inserted into a relational database;

FIG. **116** illustrates a flowchart for a method for efficiently retrieving data in accordance with an embodiment of the present invention;

FIG. **117** illustrates the manner in which a client requests information from server objects via a network;

FIG. **118** illustrates the method of the present invention in which a client requests attributes from a server object via a network;

FIG. **119** illustrates the transmitting of all data in a Data Structure from a client to a server and visa-versa;

FIG. **120** illustrates the method in which a client finds and instantiates a Customer Object from a customer component;

FIG. **121** illustrates a Structure Based Communication that builds upon the method of FIG. **120** and depicts the flow of control during Structure Based Communication;

FIG. **122** shows Five Styles of Client/Server Computing;

FIG. **123** illustrates a flowchart for a method for providing an activity module in accordance with an embodiment of the present invention;

FIG. **124** illustrates multiple interfaces to an application including a handheld device, a desktop PC, and a telecommunications device;

FIG. **125** illustrates an activity entity relationship diagram;

FIG. **126** illustrates a roles and responsibilities diagram;

FIG. **127** illustrates a typical implementation between a user interface and its activity;

FIG. **128** illustrates a flowchart for a method for structuring validation rules to be applied to a user interface for maximum maintainability and extensibility in accordance with an embodiment of the present invention;

FIG. **129** illustrates widgets with their validation requirements;

FIG. **130** illustrates a user interface validator association diagram;

FIG. **131** illustrates a validation rule class diagram;

FIG. **132** illustrates a rule validation interaction diagram;

FIG. **133** illustrates a flowchart for a method for assigning a view to an activity in accordance with an embodiment of the present invention;

FIG. **134** illustrates a manner in which the maintain customer activity operation of the present invention launches its view;

FIG. **135** illustrates the view configurer launching the maintain customer view operation;

FIG. **136** illustrates a flowchart for a method for testing successfulness of an operation having pre-conditions and post-conditions that must be satisfied for the operation to be successful in accordance with an embodiment of the present invention;

FIG. **137** illustrates an operation diagram depicting an example of pre-conditions and post-conditions;

FIG. **138** illustrates a flowchart for a method for detecting an orphaned server context in accordance with an embodiment of the present invention;

FIG. **139** illustrates a Client I that has instantiated A and C, deletes C but fails to delete A;

FIG. **140** illustrates a GarbageCollector requesting for interest in context A;

FIG. **141** illustrates a GarbageCollector requesting for interest in context B;

FIG. **142** illustrates a flowchart for a method for creating a common interface for exception handling in accordance with an embodiment of the present invention;

FIG. **143** illustrates how having many different exception types will cause the exception handling logic to grow;

FIG. **144** illustrates that groupings are not always exclusive;

FIG. **145** illustrates a flowchart for a method for recording exception handling requirements for maintaining a consistent error handling approach in accordance with an embodiment of the present invention;

FIG. **146** illustrates a flowchart for a method for minimizing the amount of changes that need to be made to exception handling logic when new exceptions are added in accordance with an embodiment of the present invention;

FIG. **147** depicts a program (i.e., the exception handler of the present invention) with a few try-catch blocks;

FIG. **148** depicts a program (the polymorphic exception handler) with smaller catch blocks;

FIG. **149** illustrates a flowchart for a method for distributing incoming requests amongst server components for optimizing usage of resources in accordance with an embodiment of the present invention;

FIG. **150** illustrates server components receiving service requests;

FIG. **151** illustrates a load balancer mediating the requests of FIG. **150**;

FIG. **152** illustrates a flowchart for a method for maintaining a security profile throughout nested service invocations on distributed components in accordance with an embodiment of the present invention;

FIG. **153** illustrates a component interaction diagram showing an interaction between a number of components in a financial system;

FIG. **154** illustrates a user manger/user context relationship diagram;

FIG. **155** illustrates a flowchart for a method for translating an object attribute to and from a database value in accordance with an embodiment of the present invention;

FIG. **156** illustrates that an attribute cannot be saved directly into the persistent store;

FIG. **157** illustrates the use of an Attribute Converter to save an attribute into a database;

FIG. **158** illustrates a flowchart for a method for controlling data in accordance with an embodiment of the present invention;

FIG. **159** illustrates the data retrieval mechanism calls being placed directly within the domain object;

FIG. **160** shows the interrelationship between a database, a persist, and an account;

FIG. **161** illustrates that the database retrieval mechanism is separated from the business object by encapsulating the logic within a data handler;

FIG. **162** illustrates the TiPersistenceStream and TiMapper of an embodiment of the present invention;

FIG. **163** illustrates a flowchart for a method for organizing data access among a plurality of business entities in accordance with an embodiment of the present invention;

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 129 of 281

FIG. **164** illustrates retrieving data piecemeal;

FIG. **165** illustrates the manner in which the present invention retrieves whole objects;

FIG. **166** illustrates a flowchart for a method for retrieving multiple business objects across a network in one access operation in accordance with an embodiment of the present invention;

FIG. **167** illustrates an example of an implementation of the Multi-Fetch Object;

FIG. **168** illustrates the Fetching of a Household object along with the other related objects using the multi object fetch results;

FIG. **169** is an interaction diagram showing when the multi object fetch is not used;

FIG. **170** illustrates a flowchart for a method for implementing an association of business objects without retrieving the business objects from a database on which the business objects are stored in accordance with an embodiment of the present invention;

FIG. **171** illustrates a flowchart for a method for mapping of retrieved data into objects in accordance with an embodiment of the present invention;

FIG. **172** illustrates an Object Identity Cache in accordance with one embodiment of the present invention;

FIG. **173** illustrates a flowchart for a method for separating logic and data access concerns during development of a persistent object for insulating development of business logic from development of data access routine in accordance with an embodiment of the present invention;

FIG. **174** illustrates a flowchart for a method for providing a warning upon retrieval of objects that are incomplete in accordance with an embodiment of the present invention;

FIG. **175** illustrates an Entity-Based Data Access System;

FIG. **176** illustrates a Retrieving Data Piecemeal System;

FIG. **177** illustrates a Commit and Rollback routine;

FIG. **178** illustrates Nested Logical Units of Work;

FIG. **179** illustrates a flowchart for a method for allowing a batched request to indicate that it depends on the response to another request in accordance with an embodiment of the present invention;

FIG. **180** illustrates a Batching Retrievals and Dependency;

FIG. **181** illustrates the Dynamically Setting Dependency;

FIG. **182** illustrates a flowchart for a method for sending a single message to all objects in a logical unit of work in accordance with an embodiment of the present invention;

FIG. **183** illustrates a Hand-crafted Message Forwarding scheme;

FIG. **184** illustrates a Generic Message Forwarding feature;

FIG. **185** illustrates a flowchart for a method for batching logical requests for reducing network traffic in accordance with an embodiment of the present invention;

FIG. **186** illustrates the manner in which the present invention sends requests independently;

FIG. **187** illustrates a manner in which the present invention registers requests;

FIG. **188** illustrates a flowchart for a method for sorting requests that are being unbatched from a batched message in accordance with an embodiment of the present invention;

FIG. **189** illustrates an Ad Hoc Registration feature;

FIG. **190** illustrates a manner in which the present invention sorts requests by weight;

FIG. **191** illustrates a flowchart for a method for assigning independent copies of business data to concurrent logical units of work for helping prevent the logical units of work from interfering with each other in accordance with an embodiment of the present invention;

FIG. **192** illustrates the MVC Implementation with Global Model;

FIG. **193** illustrates the Separate Models for Separate Business LUWs;

FIG. **194** illustrates the Canceling of one LUW Independently of Another LUW; and

FIG. **195** illustrates the Context Copying Protects Context Boundaries.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A preferred embodiment of a system in accordance with the present invention is preferably practiced in the context of a personal computer such as an IBM compatible personal computer, Apple Macintosh computer or UNIX based workstation. A representative hardware environment is depicted in FIG. **1**, which illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit **110**, such as a microprocessor, and a number of other units interconnected via a system bus **112**. The workstation shown in FIG. **1** includes a Random Access Memory (RAM) **114**, Read Only Memory (ROM) **116**, an I/O adapter **118** for connecting peripheral devices such as disk storage units **120** to the bus **112**, a user interface adapter **122** for connecting a keyboard **124**, a mouse **126**, a speaker **128**, a microphone **132**, and/or other user interface devices such as a touch screen (not shown) to the bus **112**, communication adapter **134** for connecting the workstation to a communication network (e.g., a data processing network) and a display adapter **136** for connecting the bus **112** to a display device **138**. The workstation typically has resident thereon an operating system such as the Microsoft Windows NT or Windows/95 Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. Those skilled in the art will appreciate that the present invention may also be implemented on platforms and operating systems other than those mentioned.

A preferred embodiment is written using JAVA, C, and the C++ language and utilizes object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to develop complex applications. As OOP moves toward the mainstream of software design and development, various software solutions require adaptation to make use of the benefits of OOP. A need exists for these principles of OOP to be applied to a messaging interface of an electronic messaging system such that a set of OOP classes and objects for the messaging interface can be provided.

OOP is a process of developing computer software using objects, including the steps of analyzing the problem, designing the system, and constructing the program. An object is a software package that contains both data and a collection of related structures and procedures. Since it contains both data and a collection of structures and procedures, it can be visualized as a self-sufficient component that does not require other additional structures, procedures or data to perform its specific task. OOP, therefore, views a computer program as a collection of largely autonomous components, called objects, each of which is responsible for a specific task. This concept of packaging data,

structures, and procedures together in one component or module is called encapsulation.

In general, OOP components are reusable software modules which present an interface that conforms to an object model and which are accessed at run-time through a component integration architecture. A component integration architecture is a set of architecture mechanisms which allow software modules in different process spaces to utilize each others capabilities or functions. This is generally done by assuming a common component object model on which to build the architecture. It is worthwhile to differentiate between an object and a class of objects at this point. An object is a single instance of the class of objects, which is often just called a class. A class of objects can be viewed as a blueprint, from which many objects can be formed.

OOP allows the programmer to create an object that is a part of another object. For example, the object representing a piston engine is said to have a composition-relationship with the object representing a piston. In reality, a piston engine comprises a piston, valves and many other components; the fact that a piston is an element of a piston engine can be logically and semantically represented in OOP by two objects.

OOP also allows creation of an object that "depends from" another object. If there are two objects, one representing a piston engine and the other representing a piston engine wherein the piston is made of ceramic, then the relationship between the two objects is not that of composition. A ceramic piston engine does not make up a piston engine. Rather it is merely one kind of piston engine that has one more limitation than the piston engine; its piston is made of ceramic. In this case, the object representing the ceramic piston engine is called a derived object, and it inherits all of the aspects of the object representing the piston engine and adds further limitation or detail to it. The object representing the ceramic piston engine "depends from" the object representing the piston engine. The relationship between these objects is called inheritance.

When the object or class representing the ceramic piston engine inherits all of the aspects of the objects representing the piston engine, it inherits the thermal characteristics of a standard piston defined in the piston engine class. However, the ceramic piston engine object overrides these ceramic specific thermal characteristics, which are typically different from those associated with a metal piston. It skips over the original and uses new functions related to ceramic pistons. Different kinds of piston engines have different characteristics, but may have the same underlying functions associated with it (e.g., how many pistons in the engine, ignition sequences, lubrication, etc.). To access each of these functions in any piston engine object, a programmer would call the same functions with the same names, but each type of piston engine may have different/overriding implementations of functions behind the same name. This ability to hide different implementations of a function behind the same name is called polymorphism and it greatly simplifies communication among objects.

With the concepts of composition-relationship, encapsulation, inheritance and polymorphism, an object can represent just about anything in the real world. In fact, one's logical perception of the reality is the only limit on determining the kinds of things that can become objects in object-oriented software. Some typical categories are as follows:

Objects can represent physical objects, such as automobiles in a traffic-flow simulation, electrical components in a circuit-design program, countries in an economics model, or aircraft in an air-traffic-control system.

Objects can represent elements of the computer-user environment such as windows, menus or graphics objects.

An object can represent an inventory, such as a personnel file or a table of the latitudes and longitudes of cities.

An object can represent user-defined data types such as time, angles, and complex numbers, or points on the plane.

With this enormous capability of an object to represent just about any logically separable matters, OOP allows the software developer to design and implement a computer program that is a model of some aspects of reality, whether that reality is a physical entity, a process, a system, or a composition of matter. Since the object can represent anything, the software developer can create an object which can be used as a component in a larger software project in the future.

If 90% of a new OOP software program consists of proven, existing components made from preexisting reusable objects, then only the remaining 10% of the new software project has to be written and tested from scratch. Since 90% already came from an inventory of extensively tested reusable objects, the potential domain from which an error could originate is 10% of the program. As a result, OOP enables software developers to build objects out of other, previously built objects.

This process closely resembles complex machinery being built out of assemblies and sub-assemblies. OOP technology, therefore, makes software engineering more like hardware engineering in that software is built from existing components, which are available to the developer as objects. All this adds up to an improved quality of the software as well as an increased speed of its development.

Programming languages are beginning to fully support the OOP principles, such as encapsulation, inheritance, polymorphism, and composition-relationship. With the advent of the C++ language, many commercial software developers have embraced OOP. C++ is an OOP language that offers a fast, machine-executable code. Furthermore, C++ is suitable for both commercial-application and systems-programming projects. For now, C++ appears to be the most popular choice among many OOP programmers, but there is a host of other OOP languages, such as Smalltalk, Common Lisp Object System (CLOS), and Eiffel. Additionally, OOP capabilities are being added to more traditional popular computer programming languages such as Pascal.

The benefits of object classes can be summarized, as follows:

Objects and their corresponding classes break down complex programming problems into many smaller, simpler problems.

Encapsulation enforces data abstraction through the organization of data into small, independent objects that can communicate with each other. Encapsulation protects the data in an object from accidental damage, but allows other objects to interact with that data by calling the object's member functions and structures.

Subclassing and inheritance make it possible to extend and modify objects through deriving new kinds of objects from the standard classes available in the system. Thus, new capabilities are created without having to start from scratch.

Polymorphism and multiple inheritance make it possible for different programmers to mix and match character-

istics of many different classes and create specialized objects that can still work with related objects in predictable ways.

Class hierarchies and containment hierarchies provide a flexible mechanism for modeling real-world objects and the relationships among them.

Libraries of reusable classes are useful in many situations, but they also have some limitations. For example:

Complexity. In a complex system, the class hierarchies for related classes can become extremely confusing, with many dozens or even hundreds of classes.

Flow of control. A program written with the aid of class libraries is still responsible for the flow of control (i.e., it must control the interactions among all the objects created from a particular library). The programmer has to decide which functions to call at what times for which kinds of objects.

Duplication of effort. Although class libraries allow programmers to use and reuse many small pieces of code, each programmer puts those pieces together in a different way. Two different programmers can use the same set of class libraries to write two programs that do exactly the same thing but whose internal structure (i.e., design) may be quite different, depending on hundreds of small decisions each programmer makes along the way. Inevitably, similar pieces of code end up doing similar things in slightly different ways and do not work as well together as they should.

Class libraries are very flexible. As programs grow more complex, more programmers are forced to reinvent basic solutions to basic problems over and over again. A relatively new extension of the class library concept is to have a framework of class libraries. This framework is more complex and consists of significant collections of collaborating classes that capture both the small scale patterns and major mechanisms that implement the common requirements and design in a specific application domain. They were first developed to free application programmers from the chores involved in displaying menus, windows, dialog boxes, and other standard user interface elements for personal computers.

Frameworks also represent a change in the way programmers think about the interaction between the code they write and code written by others. In the early days of procedural programming, the programmer called libraries provided by the operating system to perform certain tasks, but basically the program executed down the page from start to finish, and the programmer was solely responsible for the flow of control. This was appropriate for printing out paychecks, calculating a mathematical table, or solving other problems with a program that executed in just one way.

The development of graphical user interfaces began to turn this procedural programming arrangement inside out. These interfaces allow the user, rather than program logic, to drive the program and decide when certain actions should be performed. Today, most personal computer software accomplishes this by means of an event loop which monitors the mouse, keyboard, and other sources of external events and calls the appropriate parts of the programmer's code according to actions that the user performs. The programmer no longer determines the order in which events occur. Instead, a program is divided into separate pieces that are called at unpredictable times and in an unpredictable order. By relinquishing control in this way to users, the developer creates a program that is much easier to use. Nevertheless, individual pieces of the program written by the developer still call libraries provided by the operating system to accomplish certain tasks, and the programmer must still determine the flow of control within each piece after it's called by the event loop. Application code still "sits on top of" the system.

Even event loop programs require programmers to write a lot of code that should not need to be written separately for every application. The concept of an application framework carries the event loop concept further. Instead of dealing with all the nuts and bolts of constructing basic menus, windows, and dialog boxes and then making these things all work together, programmers using application frameworks start with working application code and basic user interface elements in place. Subsequently, they build from there by replacing some of the generic capabilities of the framework with the specific capabilities of the intended application.

Application frameworks reduce the total amount of code that a programmer has to write from scratch. However, because the framework is really a generic application that displays windows, supports copy and paste, and so on, the programmer can also relinquish control to a greater degree than event loop programs permit. The framework code takes care of almost all event handling and flow of control, and the programmer's code is called only when the framework needs it (e.g., to create or manipulate a proprietary data structure).

A programmer writing a framework program not only relinquishes control to the user (as is also true for event loop programs), but also relinquishes the detailed flow of control within the program to the framework. This approach allows the creation of more complex systems that work together in interesting ways, as opposed to isolated programs, having custom code, being created over and over again for similar problems.

Thus, as is explained above, a framework basically is a collection of cooperating classes that make up a reusable design solution for a given problem domain. It typically includes objects that provide default behavior (e.g., for menus and windows), and programmers use it by inheriting some of that default behavior and overriding other behavior so that the framework calls application code at the appropriate times.

There are three main differences between frameworks and class libraries:

Behavior versus protocol. Class libraries are essentially collections of behaviors that you can call when you want those individual behaviors in your program. A framework, on the other hand, provides not only behavior but also the protocol or set of rules that govern the ways in which behaviors can be combined, including rules for what a programmer is supposed to provide versus what the framework provides.

Call versus override. With a class library, the code the programmer instantiates objects and calls their member functions. It's possible to instantiate and call objects in the same way with a framework (i.e., to treat the framework as a class library), but to take full advantage of a framework's reusable design, a programmer typically writes code that overrides and is called by the framework. The framework manages the flow of control among its objects. Writing a program involves dividing responsibilities among the various pieces of software that are called by the framework rather than specifying how the different pieces should work together.

Implementation versus design. With class libraries, programmers reuse only implementations, whereas with frameworks, they reuse design. A framework embodies the

way a family of related programs or pieces of software work. It represents a generic design solution that can be adapted to a variety of specific problems in a given domain. For example, a single framework can embody the way a user interface works, even though two different user interfaces created with the same framework might solve quite different interface problems.

Thus, through the development of frameworks for solutions to various problems and programming tasks, significant reductions in the design and development effort for software can be achieved. A preferred embodiment of the invention utilizes HyperText Markup Language (HTML) to implement documents on the Internet together with a general-purpose secure communication protocol for a transport medium between the client and the Newco. HTTP or other protocols could be readily substituted for HTML without undue experimentation. Information on these products is available in T. Berners-Lee, D. Connoly, "RFC 1866: Hypertext Markup Language—2.0" (November 1995); and R. Fielding, H, Frystyk, T. Bemers-Lee, J. Gettys and J. C. Mogul, "Hypertext Transfer Protocol—HTTP/1.1: HTTP Working Group Internet Draft" (May 2, 1996). HTML is a simple data format used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of domains. HTML has been in use by the World-Wide Web global information initiative since 1990. HTML is an application of ISO Standard 8879; 1986 Information Processing Text and Office Systems; Standard Generalized Markup Language (SGML).

To date, Web development tools have been limited in their ability to create dynamic Web applications which span from client to server and interoperate with existing computing resources. Until recently, HTML has been the dominant technology used in development of Web-based solutions. However, HTML has proven to be inadequate in the following areas:

- Poor performance;
- Restricted user interface capabilities;
- Can only produce static Web pages;
- Lack of interoperability with existing applications and data; and
- Inability to scale.

Sun Microsystem's Java language solves many of the client-side problems by:

- Improving performance on the client side;
- Enabling the creation of dynamic, real-time Web applications; and
- Providing the ability to create a wide variety of user interface components.

With Java, developers can create robust User Interface (UI) components. Custom "widgets" (e.g., real-time stock tickers, animated icons, etc.) can be created, and client-side performance is improved. Unlike HTML, Java supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance. Dynamic, real-time Web pages can be created. Using the above-mentioned custom UI components, dynamic Web pages can also be created.

Sun's Java language has emerged as an industry-recognized language for "programming the Internet." Sun defines Java as: "a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, dynamic, buzzword-compliant, general-purpose programming language. Java supports programming for the Internet in the form of platform-independent Java applets." Java applets are small, specialized applications that comply with Sun's Java Application Programming Interface (API) allowing developers to add "interactive content" to Web documents (e.g., simple animations, page adornments, basic games, etc.). Applets execute within a Java-compatible browser (e.g., Netscape Navigator) by copying code from the server to client. From a language standpoint, Java's core feature set is based on C++. Sun's Java literature states that Java is basically, "C++ with extensions from Objective C for more dynamic method resolution."

Another technology that provides similar function to JAVA is provided by Microsoft and ActiveX Technologies, to give developers and Web designers wherewithal to build dynamic content for the Internet and personal computers. ActiveX includes tools for developing animation, 3-D virtual reality, video and other multimedia content. The tools use Internet standards, work on multiple platforms, and are being supported by over 100 companies. The group's building blocks are called ActiveX Controls, small, fast components that enable developers to embed parts of software in hypertext markup language (HTML) pages. ActiveX Controls work with a variety of programming languages including Microsoft Visual C++, Borland Delphi, Microsoft Visual Basic programming system and, in the future, Microsoft's development tool for Java, code named "Jakarta." ActiveX Technologies also includes ActiveX Server Framework, allowing developers to create server applications. One of ordinary skill in the art readily recognizes that ActiveX could be substituted for JAVA without undue experimentation to practice the invention.

OVERVIEW

ARCHITECTURE BASICS

Architecture Overview

What is Architecture?

Architecture—whether the word is applied to work with a city skyline or an information system—is both about designing something and about making, building, or constructing something. An architect is literally a "master builder"—from the Greek words archi (primary or master) and tekton (builder or carpenter). In good Greek fashion, however, it would be unthinkable for something to be built without a sound theoretical basis. So architecture involves theory, but there is nothing merely theoretical about it. Conversely, architecture is also eminently practical, but there is nothing merely practical about it. Ideas about form and structure lie behind architecture. Ultimately one must let go of a mindset that tries to separate the designing from the making; they exist together as a whole, and to extract one without the other is to kill the whole.

Architecture also is an engineering discipline. It creates and also depends on a structured manner to analyze and design whatever is to be built. Like all living disciplines, architecture continues to grow and evolve. Engineering discoveries move the field forward. Certain design and engineering principles clearly show themselves to be successful in practice, and these then become repeatable components of additional work. The ability to continue to master each component, as well as the interrelations among components, is a distinguishing characteristic of architecture.

So architecture is about designing and building something from a set of basic components, and also about the interrelations among the components. And it is a discipline

whereby all these things come together—materials, space, people—to bring something into being that was not there before.

Although building architects have not always been pleased about it, architectural concepts have influenced other kinds of "building" projects for some time. Over the past twenty years, developers of information systems, for example, have used concepts from the field of architecture not only to describe their work but to execute it, as well.

The use of architectural thinking implies that the work is about creating certain kinds of structures that can be engineered or at least influenced, and that the work can be organized and performed in a structured, systematic manner. Moreover, use of architectural concepts implies that there is something repeatable about the work: architects can create a structure, then use components of that structure again in the future when they come across a similar situation.

An architectural paradigm should not be lightly used. It makes demands. To use architectural concepts implies that clients are ready to do so—that is, that the field is sufficiently mature in its work to see patterns and to organize future work according to those patterns.

Finally, architecture must be understood as a process **200**, not just a thing. This process can be described at a very high level using FIG. **2**.

Step **1**: Analyze **202**. The architect must begin by listening to and researching the needs of the client. What is the function of the building? What is its environment? What are the limitations set by budget and use?

Step **2**: Design **204**. This is a blueprint stage. The architect creates one or several designs showing the layout of the structure, how different spaces fit together, how everything looks from different views, what materials are to be used, and so forth.

Step **3**: Model & Test **206**. Not every architectural project has this step, but in many cases, the architect will create a scale model/prototype of the finished product, allowing the client a clearer sense of what the ultimate solution will look like. A model is a kind of test stage, allowing everyone to test the design in a near-real-life setting.

Step **4**: Build **208**. This is the actual construction of the building, in general accord with the blueprints and prototype.

Step **5**: Operate and Evolve **210**. The building is to be lived in and used, of course, and so an important step is to ensure that the finished product is tended and operated effectively. Architects themselves may not be involved in the operation of their building, but they certainly would be involved in future expansions or evolutions of the building. Stewart Brand's recent text, How Buildings Learn, argues that effective architecture takes into account the fact that buildings "learn": as people live and work in them over time, those people will seek to alter the building in subtle, or not so subtle, ways.

Also, when architects design a building, they have in their heads a primary conceptual framework for all the components that go into that building: the plumbing, the electric, the sewers, stairs/elevators, framing structure, and so forth. The tacit step for an architect is, "Based on my knowledge of the generic components that go into a building, how will these components fit together in this particular building? Which of these components will require special attention because of the functional demands of the building?"

Oxford English Dictionary Definition

The conceptual structure and overall logical organization of a computer or computer-based system from the point of view of its use or design; a particular realization of this.

Gartner Group Definition

The manner or structure in which hardware or software is constructed. Defines how a system or program is structured, how various components and parts interact, as well as what protocols and interfaces are used for communication and cooperation between modules and components which make up the system.

Gartner Group sets forth seven general characteristics of successful architectures.

Delimitation of the problem to be addressed

Decomposition of the solution to components with clearly assigned responsibilities

Definition of interfaces, formats, and protocols to be used between the components. These should be sufficiently clear and robust in order to permit asynchronous development and ongoing re-implementation of the components.

Adequate documentation to permit compliance by implementors

An auditing mechanism that exercises the specified interfaces to verify that specified inputs to components yield specified results

An extendibility mechanism to enable response to changing requirements and technologies

Policies, practices, and organizational structures that facilitate adoption of the architecture

What Types of Architectures are Discussed in the Following Description?

Standard Architecture Framework (SAF) **300** provides access to the user's thought leadership and architecture frameworks for Execution, Development and Operations environments **302**,**304**,**306**. For a more detailed discussion on these architectures, please see Standard Architecture Summaries (below). FIG. **3** shows the dependencies of the three architecture frameworks and is described in more detail in the Delivery Vehicle Overview (below).

The following lists are starting points for considering the range of components and activities that must be covered by each architectural view of the system. They are not a definitions of the environments.

Standard Architecture Summaries

Execution Architecture **302**

The execution architecture is a unified collection of run-time technology services, control structures, and supporting infrastructure upon which application software runs.

It includes components such as:

Application messaging

Batch processing architecture

Middleware

Reporting

Error handling

On-line architecture

Security

Code/decode

Data access methods

Integrated help

File transfer capabilities

Directory services

Load balancing

Workflow services

State management

"Special" requirements (e.g., workflow, telephony, groupware)

Development Architecture Framework **304**

The Development Architecture Framework (DAF) is a unified collection of technology services, tools, techniques, and standards for constructing and maintaining application software.

It includes components such as:

Design/documentation tools

Information repository

Project Management tools

Program Shells

GUI Window painter

Prototyping tools

Programmer APIs

Testing tools

Source code control/build process

Performance test tools

Productivity tools

Design tools

Compiler/debugger

Editor

Refer to the Development Architecture Framework application (referenced above) for more information.

Operations Architecture **306**

A unified collection of technology services, tools, standards and controls required to keep a business application production or development environment operating at the designed service level. It differs from an execution architecture in that its primary users are system administrators and production support personnel.

It includes components such as:

Job scheduler

Software distribution

Error monitor

Data backup and restore

Help desk

Security administration

High-Availability

Hardware management

Performance monitors

Startup/shutdown procedures

Report management tool

Disaster Recovery

Network Monitoring Tools

Cross Platform Management Tools

Considerations—all environments

To ensure that you are asking the right questions about the technology architecture, you must refer to the Architecture Checklist (available from the Content Finder). Questions will include:

For all technology components, have the following characteristics been addressed:

Performance according to specifications?

Reliability of operation?

Ease of operation?

Maintenance requirements?

Ability to interface with other components, particularly those from other vendors?

Delivery schedule to provide adequate pre-conversion testing?

Backup procedures?

Vendor reliability and financial stability?

Future proofing against business change?

Have the versions of system software been live at another site for at least six to twelve months?

This time frame varies by product. Have reference sites been verified?

What is a Framework?

It is a major challenge to design the complex infrastructure that is needed to satisfy the requirements of today's distributed, mission-critical applications. As such, it is helpful to have an inventory of the components that may be required for the design, build, installation and operation of systems. It is also helpful to have an understanding of how the components fit together conceptually.

A Framework should be thought of as a conceptual structure used to frame the work about to be done. It should be used as a thought trigger or as a completeness check. You cannot build from a framework directly but instead should use it as a starting point for understanding and designing.

Frameworks are used to help practitioners understand what components may be required and how the components fit together. Based on the inventory of components and the description of their relationships, practitioners will select the necessary components for their design. An architect extracts components from one or more Frameworks to meet a specific set of user or application requirements. Once an architecture has been implemented it is often referred to as an architecture or an infrastructure.

The scope of what a framework addresses can vary widely. One framework, for instance, may outline the components for a technical infrastructure in its entirety whereas another framework may focus explicitly on the network. A thorough understanding of a framework's scope is crucial to its use during the design phase of a project.

It is also important to understand whether the framework is vendor specific in nature (proprietary) or whether it is available for use by a large number of vendors (open).

Why is Architecture Important?

One has seen the benefits of an architectural approach to information systems development: better productivity and less reinvention of the wheel. An architecture provides a completeness check, ensuring that all relevant components of a possible solution have been considered. It ensures consistent, reliable, high-quality applications. It gives everyone—the developers and their clients—a common framework and common language with which to talk about the work.

Perhaps most important, it allows developers to leverage successful solutions when performing additional work. Architecture involves repeatable concepts, and so it reduces the time and cost by which a solution is delivered.

Some of the specific technical benefits of a good architecture are:

Simplified Application Development

Provides common set of application services. Removes application programmers from the complexities of the underlying technology and development tools, allowing less experienced developers to be more productive

Quality

Usually more experienced developers implement the often complex technical components in an architecture. These components are then reused, avoiding duplicated complex logic in the applications. Iterations during design, implementation and testing often result in refinement and improvement of the architecture components. All users of these components benefit from such improvements, reducing the risk of failure and ensuring better overall quality in the final application.

Integration

An architecture often ties together disparate software, platforms and protocols into one comprehensive framework.

Extensibility

The architecture is established by experienced personnel who can predict with some confidence whether a given architecture will fulfill current and future requirements. Code extensions are easily integrated. A well-balanced architecture consists of the "right" components, where the components are tied together by simple interrelationships, since complex relationships increase the architecture's complexity faster than modularization can reduce it.

Location Transparency

Divorces application from the details of resource location. This is however not always true or required. For performance reasons designers and developers still often need to be aware of process and data locations.

Horizontal Scaling

Assist in optimal utilization of existing infrastructure resulting in increased application performance and stability

Isolation

An architecture can be used to isolate the applications from particular products. This ensures that products can more easily be replaced later. This characteristic can be important if there is risk associated with a product's or product vendor's future, or the rate of change in a particular technology area is particularly high. An evident example is looking back at changes in past user interface standards. Applications that did not separate user interface logic from business logic, had to be completely rewritten to take advantage of new user interfaces, such as MS Windows and more recently Web browsers.

Portability

Increases portability and reusability within and across different platforms or protocols.

The use of architecture frameworks during analysis and design can reduce the risks of an IT solution. It should improve development productivity through reuse, as well as the IT solution's reliability and maintainability.

One key challenge for today's IT managers is the need for change. Architectures provide a basic framework for major change initiatives. Clients' core business is performed by strategic applications that will most likely require frequent and rapid development to handle changes in technology capability and business requirements. A properly defined and intelligently developed architecture delivers an infrastructure on which clients can build and enhance applications that support their current and future business needs. This is how one helps clients to manage change.

A key benefit of an architecture is that it divides and conquers complexity. Simple applications benefit less from architecture than complex ones do; fewer decisions are needed in these cases, and fewer people need to know about them. During maintenance, a poorly architected small application is tolerable because it is still relatively easy to locate a fault and to anticipate the side effects of correcting it. Conversely, complex applications are more difficult to understand and to modify. Complexity is reduced by subdividing the application in layers and components, each layer having a specific functionality. The layers are strongly cohesive and de-coupled: A given layer does not need to know the internals of any other layer.

The following quote from a recent study of Large Complex Systems (LCS) stress the importance of a stable architectures in large systems:

Successful delivery of an LCS solution depends on the early definition and use of common data applications and technology architecture.

There is a high failure rate when the architecture is not defined, stabilized, and delivered early in an LCS effort.

All significant LCS efforts involved the use of common or shared architectures. A successful effort, however, depended on early definition and delivery of a stable common architecture.

Significant changes to the data, application, or technology architectures had severe negative effects on the timeliness of project deliverables, and on the reliability of what was delivered.

PROJECT1 and PROJECT2, for example, experienced unusual circumstances. While the client evaluated whether to proceed, one defines and designs the architecture. As a result, the teams had nine months to define, design, and begin implementation of required data, applications, and development architectures. Although in each case these architectures continued to evolve with business and technology needs, they remained largely consistent with the initial design. This consistency proved to be essential to the timely delivery of the applications.

At PROJECT3 and PROJECT4, on the other hand, the architectures went through major evolutions as the developers created the applications. The overall result was that those efforts experienced delays relative to plan.

Although it is not realistic for every project to have nine months to define required architectures, it does suggest that early focus on definition and design of the architectural components is essential.

The risk of failure is greatly increased if essential architectures are being defined or changed significantly in parallel with application development.

What are the Benefits of an Architecture?

The benefits derived from a technology architecture may allow a user to be in the forefront of the development of many leading edge business solutions. The investment in a reliable and flexible architecture can result in one or more of the following:

Preservation of investments in applications and technology by isolating each from changes in the other (e.g. upgrades in hardware or third-party software do not impact applications).

Leveraging scarce technical skills (e.g. the need for people with detailed skills in a specific communications protocol or aspects of SQL).

Enhancements in productivity, flexibility and maintainability because common and often complex and error-prone components (e.g. error handling or cross-platform communications) are created within the architecture, and then reused by all applications.

Increases in the predictability of application performance because the run-time behavior of common components is familiar and consistent.

Serves as a construction blueprint and discussion agenda and ensures consistency across systems. This can have a big impact on the operability and maintenance of the delivered applications.

What is an Architect?

Architects must have deep understanding of a project, business and/or technical environment. Architects are involved across business integration projects, managing their complexities and intricacies.

How Advanced Should an Architect be?

It is easy to go overboard when designing and implementing a technology architecture. Ideally the architecture should be a thin, well-defined layer that ensures development productivity, maintenance flexibility, performance and stability.

A key issue is maintainability and operability. Keep in mind that others may have to understand the rationale behind the architecture design in order to correctly maintain it.

Architecture logic can quickly become very abstract and hard to maintain by others than those who built it. A carefully designed architecture can quickly be destroyed by maintenance personnel that do not understand how it was designed and developed.

You should make your architecture as light-weight as possible only addressing the requirements that drive it. Avoid "nice to have" flexibility and additional levels of abstractions that are intellectually interesting but not strictly required.

Delivery Vehicle Overview

A Delivery Vehicle is an integrated collection of technology services that supports an application style, implemented on a distinct architecture generation.

Application Style

An application style defines a unique class of processing type, which is used by applications, and thus end-users. Delivery Vehicle Reference set of Application Styles include batch, on-line transaction processing, collaboration, data warehouse, knowledge management and integration.

The Application Style is the primary dimension of a Delivery Vehicle, and most people use the terms Application Style and Delivery Vehicle to mean the same thing.

A key goal with a delivery vehicle is that it can be reused across many applications. It is still part of the Technology Architecture, not involving application specific logic. An Application Architecture on the other hand, will be specific for a particular application.

Architecture Generation

An architecture generation is a broad classification scheme for placing technology components within a technology era. Delivery Vehicles are physically implemented on a distinct architecture generation. Examples of architecture generations include host-based, client-server and net-centric.

Note: Defining a clear line between what falls under the client/server and a Netcentric technology generation is difficult; typically different people tend to have different opinions. Technologically, the Netcentric generation may be an evolution of the client/server generation. In the context of the Delivery Vehicles, the technology generation discussion may be intended to be a logical discussion that aims to highlight the new business capabilities enabled by new technologies. So for example, there could be a PowerBuilder application executing from a Web Browser using a plug-in. Whether this is called a client/server or Netcentric application is up to the reader. When presenting technology architecture information to clients, focus on the business capabilities that are offered by technologies rather than just on definitions for what is client/server or what is Netcentric technology.

Delivery Vehicle Matrix

FIG. **4** illustrates a delivery vehicle matrix **400**. One way of looking at a Delivery Vehicle is therefore as an intersection of a technology generation **402** and application style **404**. This is the presentation method currently adopted for navigation in SAF.

Delivery Vehicle Cube

The Delivery Vehicle Cube **500**, illustrated in FIG. **5**, represents the "full" picture of what a Delivery Vehicle is. In addition to the Application Styles and the Technology generations it introduces a distinction between Execution, Development and Operations Environments **502**,**504**,**506**.

The cube has the following dimensions, or cube "faces:

1. On the bottom left face of the cube are the core technology components and services **508** that are common across all delivery vehicles.

These core services may be implemented using one or several of the Technology Generations; currently Host, Client/Server or Netcentric. Most major enterprises have legacy systems that include both host based and distributed client/server applications. Netcentric applications may extend the mix of system technologies.

2. On the top left of the cube are the technology components **510** that are required to support a distinct delivery vehicle.

These components extend the technology architecture with services that are specific for each distinct delivery vehicle. Some of the components may extend some of the core services

3. On the right face of the cube are the three environments each delivery vehicle will affect: execution, development and operations **502**,**504**,**506**.

Both the core services and the delivery vehicle extensions require support in all three environments. The cube illustrates that different delivery vehicles may require different extensions to a core development or operations environment, not just the execution architecture. A mission-critical high-volume transaction delivery vehicle may require special performance tuning tools in the development architecture, as well as real-time monitoring tools in the operations architecture.

Also different technology generations may require special services in all three environments. When working in a multi-platform environment, there may be duplicated services across platforms. This usually complicates development, operations and execution architectures and may require special focus on providing an integration architecture.

The following figure illustrates the relationship between the three environments and the overall business system:

Typically, one may focus on engagements regarding the execution environment. The main dependency between these three environments is that the execution architecture to a large degree drives the requirements for the development and operations architectures. For example if a heterogeneous, distributed execution architecture is selected, both the development and operations environments must reflect this.

How can the Delivery Vehicle Framework be Useful?

Refocus users and clients toward business solutions and away from technology issues.

Help you link architecture planning deliverables to delivering.

Create an enterprise-wide view of the business capabilities enabled by technologies.

Provide new architecture frameworks needed today to meet you're a user's client's business needs.

Provide guidance to define what architecture best meets you're a user's client's business needs.

Provide standard architecture frameworks and best practices to build these architectures.

During a high-level architecture design, help the user identify architecture services the user will need to address, by providing a logical level discussion one can use to assess types of base services and products needed for the specific situation.

When Delivery Vehicles are implemented, they reduce time to implement business solutions by providing "Starter Kits" architectures.

When Delivery Vehicles are implemented, they leverages technology across the business by:

reducing operations and maintenance costs by limiting the number of different technologies and skills required to support these technologies.

reducing technology costs for execution & development.

Note: The Delivery Vehicle Framework presents a way to organize technology architecture information. When presenting this type of contentclient, one may need to tailor the information they present based on the client's background and the terminology they are familiar with.

Technology Generation Selection

Introduction

This section should assist an architect in understanding the characteristics of, and the implications from selecting, a specific technology generation. The strengths and weaknesses of each technology generation should be understood when planning and designing a system. When identifying the core technologies to be used in an architecture, a view of the client's existing IT architecture **600**, guiding principles **602** and business imperatives **604** should be taken into consideration, as depicted in FIG. **6**.

It is important to realize that a distinct, static division does not exist between the different technology generations. It is possible that an architecture may consist of components from more than one generation.

The goal should be to understand the pros and cons of the different technology options available for each component and to select the most appropriate one based on the client's requirements.

It is becoming more, important to leverage existing systems and integrate them with new applications. A typical scenario can involve mainframe legacy systems acting as servers in a client server architecture, application servers being accessed from both traditional GUI clients built in Powerbuilder and Visual Basic and from Web-based front ends accessing the application servers via a Web-server.

General Considerations

From a technology point of view a new custom-made application should generally use the most recent Architecture Generation to assure that the application will live longer by better being able to adapt to future changes.

This implies that most applications should ideally be based on a Netcentric Architecture, rather than on a traditional client/server or a host-based architecture.

However choosing a generation is not just a technical decision. Often key technology architecture decisions are made as a result of factors which are completely non-technical in nature, such as financial factors, internal and client politics (say no more), and implementation/operational considerations.

When deciding whether to employ a Netcentric solution, i.e. incorporating Web-based user interfaces and Internet application styles, keep in mind that these technologies are not a panacea and should be used only when there is solid business reason. They require new investments in skills, tools, development and operations processes. Due to the relative immaturity of tools and products, they also represent additional risks both in technical terms, such as performance and reliability, and in strategic terms, such as vendor and product quality and stability.

Regardless today each project should always consider the prospect of utilizing Netcentric technologies. It is important to evaluate whether the application can benefit from a Netcentric style implementation immediately or in the future.

Even if a traditional client/server approach (e.g. using Visual Basic or PowerBuilder) is decided upon, the use of Netcentric concepts to produce significant reductions in software packaging and distribution costs should be considered. Such concepts include three- or multi-tier architectures with more business logic residing on server, flexible security architecture, and user interface concepts that can be ported to a Web Browser at a later stage.

A Netcentric architecture will usually still support development of client/server applications. The opposite is not often true since traditional client/server systems usually keep a substantial portion of the business logic on a fat client, while Netcentric architectures still favor keeping most business logic at the server side. Also Netcentric architectures tend to be more loosely coupled than (the still dominant two-tier) client/server systems.

The following sections identify the main characteristics associated with a Netcentric, Client Server or Host based technology generation. This list should in no way be considered complete and exhaustive but is included as a starting point from which the identification process may begin.

Network Centric Architecture Generation

If, based upon one's client's requirements, most of the statements in FIG. **7** are true, one should consider an application based upon the Netcentric technology generation.

The following details the importance of each of the statements in FIG. **7** and should assist one in identifying the appropriate answer for the specific client engagement.

Existing Architecture and Infrastructure **700**

E**1**. Other Netcentric applications been developed and placed in production.

The user community is often less resistant to accept the use of new technology to address changing business drivers if they are not completely unfamiliar with the characteristics of the technology. If an application based on a Netcentric architecture has already been successfully piloted or deployed, acceptance of additional systems will be eased.

E**2**. The client has significant technology skills within its IT department.

This is especially important if the client plans on developing or operating the application themselves. A significant investment in training and changes to internal organizations may be necessary for successful deployment of this type of system. The client must have a culture that supports change. Some organizations are very conservative and strong, making it difficult to deliver a successful project using new technology.

E**3**. The client has multiple hardware/operating system configurations for their client machines.

In traditional client/server environments, distributing an application internally or externally for an enterprise requires that the application be ported, recompiled and

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 138 of 281

tested for all specific workstation operating systems. Use of a Universal Client or web-browser may eliminate many of these problems by providing a consistent and familiar user interface on many different operating systems and hardware platforms.

E4. The application will run on a device other than a PC.

The momentum of the Internet is putting a lot of pressure on vendors of various devices to be web-enabled. Having the Internet infrastructure in place makes it more feasible for vendors to create new physical devices from which electronic information can be accessed. For example, Web televisions are gaining momentum. Now users can access the Internet from a television set. Network Computers, thin-client devices that download and run applications from a centrally maintained server are generating a lot of interest. Also, users want to have access to the same information from multiple physical devices. For example, a user might want to have access to his/her e-mail from a cellular phone, from a Web TV or their portable PC.

E5. The current legacy systems can scale to serve a potentially large new audience.

Expanding the user community of a legacy host or client/server system by including an audience which is external to the company can result in dramatic increases in system usage. The additional demand and increased usage placed on existing legacy systems is often difficult to estimate or predict. Analysis must be conducted to ensure existing legacy systems and infrastructure can absorb this increase.

Business Imperatives **702**

B1. The client needs to reach a new external audience with this application.

This is probably the main reason for selecting a Netcentric architecture. Through appropriate use of a Netcentric architecture it is often possible to gain exposure, to new customers and markets. The client can often achieve significant competitive advantage by providing new services and products to its customers. Also this new channel makes it technically possible to develop a new generation of "market-of-one" products, where each customer can repeatedly and easy customize a product according to own preferences.

B2. The client needs to reach a large or diverse internal audience with this application.

Configuration management of traditional client/server applications, which tend to be physically distributed across both the client and server, is a major issue for many corporations. The software distribution of such applications which are packaged as one large or a combination of a few large executables makes minor updates difficult for even a small scale user population. Every time an update is made, a process must be initiated to distribute new code to all client machines. The browser-centric application style offers an alternative to this traditional problem of distributing functionality to both internal and external users.

IT Guiding Principles **704**

G1. The client is an early adopter of new technology.

Implementation of a Netcentric architecture can help the client realize a number of business benefits. However, the introduction of new technology into an organization does have inherent risks and can result in a significant amount of change. The client should have a culture which can embrace these necessary changes.

G2. Applications should be developed to handle non-dedicated or occasional users.

Non-expert users need a simple to use and familiar interface in order to be able to use the application. As people grow accustomed to Web-browsers, this will be their preferred user-interface. The consistent interface provided by the Web-browsers will help reduce the learning curve necessary for becoming familiar with new applications.

G3. Where appropriate, applications should be developed with multi-media capabilities for the presentation of data (text, sound, video, etc.).

The ability to digitize, organize, and deliver textual, graphical and other information (e.g., video, audio, etc.) in addition to traditional data to a broader audience, enables new methods for people and enterprises to work together. Netcentric technologies (e.g., HTML documents, plug-ins, Java, etc.) and standardization of media information formats enable support for these types of complex documents and applications. Network bandwidth remains a performance issue. However advances in network technologies and compression techniques continue to make richer media-enabled documents and applications more feasible on the Web.

G4. The Execution, Operation and Development architectures will be designed to support frequent releases of enhancements/modifications to production applications.

It is imperative that companies in the current market place be able to quickly modify their business processes in order to address changes in the industry.

A Netcentric architecture simplifies frequent software releases for both internal and external users of the systems.

Client/Server Network Generation

If, based upon a client's requirements, most of the statements of FIG. **8** are true, one should consider an application based upon the Client Server technology generation.

The following section details the importance of each of the statements found in FIG. **8** and should assist one in identifying the appropriate answer for your specific client engagement.

Existing Architecture and Infrastructure **800**

E1. Other Client Server applications been developed and placed in production and the client IT organization contains personnel familiar with client server architecture concepts.

As with any new technology, there is a learning curve related to attaining client server development skills. The development process is often much more efficient when familiar tools and environments are used. The introduction of new technology can also create instability in the operations environment. Client/server systems still represent a new technology to many IT departments.

Business Imperatives **802**

B1. The application will be used only by an internal user community.

Software distribution is a concern for traditional client server computing environments due to the fact that executable and data files need to reside on the client hard drive. Distribution to a user community outside of the client's organization is even more difficult to implement and manage and will probably be limited to a few key business partners.

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 139 of 281

B2. The application requires an advanced, dynamic, and integrated user interface for expert users.

State of the art 4GL and 3GL development languages will support advanced user interfaces which require a significant degree of context management between fields and windows. Web-based user interfaces do not support such interfaces well yet.

B3. Session performance is critical to the application or sub-second response times are required for successful use.

Client server applications can provide response times necessary to support transaction intensive mission critical systems. Application logic and business data can be distributed between the client and server for optimal efficiency. Web-based interfaces still have an inherent overhead due to the connectionless communication and constant downloading of data, formatting information and applet code.

B4. The application needs to support off-line mobile users.

Mobile computing is becoming more prevalent in the work place, therefore, connectivity to a server can not be assumed for all user classes. A client server architecture allows for the distribution of application logic and/or data between the server and client. Replication of data and logic is usually necessary for applications that are run on portable computers.

IT Guiding Principles **804**

G1. The client maintains their applications internally and the IT department has the necessary resources, organizations and processes to maintain a Client Server application.

Introduction of a Client Server application to a company's production environment can require a great deal of change to the Execution, Operations and Development architectures required to develop, run and support the production systems. Before a Client Server application is developed, it is important that the client identify how a system of this type will fit within the company's strategic technology plan.

Host Architecture Generation

If yclients business and technical requirements meet the following system characteristics, you should consider an application based upon the Host technology generation.

The following section details the importance of each of the statements found in FIG. **9** and should assist you in identifying the appropriate answer for your specific client engagement.

Existing Architecture and Infrastructure **900**

E1. The client currently maintains and operates host based applications; and the IT organization contains personnel familiar with the development and operation of these types of applications.

Few organizations introduce solely host based production systems. Usually the infrastructure for this type of systems already exists. New development is uncommon, typically existing legacy systems need to be extended.

Host systems usually have a mature and stable operations environment. Note that mainframe expertise may be expensive and in high demand

Business Imperatives **902**

B1. The application will only be used by a dedicated, expert user community where a GUI is not needed.

A dedicated work force with low turnaround, skilled in the use of character based 3270 applications, eliminates the need for a GUI interface.

B2. The application requires a high volume of repetitive transactions.

The high degree of processing power provided by mainframes allows for the development of applications with very high performance requirements.

B3. The application has a requirement for significant batch processing.

Mainframes are probably still the most powerful platforms for large scale batch processing. Mature tools exist for scheduling, recovery/restart, sorting, merging, and moving large sets of data.

B4. End users can maintain a physical connection to the host at all times.

Physical connection to the host is required for use of the applications. Methods of mobile computing with distribution of data or business logic is not possible.

B5. The application will need to support a large number of users (>1000).

The processing power of today's mainframe lends itself well to the development of large scale, mission critical applications with a large user base.

IP Guiding Principles **904**

G1. The Client has the resources, organizations and processes necessary for the development and operation of a Host based application.

Before a Host based application is developed, it is important that the client identify how a system of this type will fit within the company's strategic technology plan.

G2. Reliance upon a single vendor (IBM) for technology solutions is acceptable.

Selection of a host based architecture inherently locks the client into dependence upon one vendor for its technology solutions. While IBM is a reputable, stable company it may be important to ensure that the client's long term business strategy will be supported by IBM's technology vision and direction.

G3. Centralized application and data is an acceptable strategy.

A pure host based architecture eliminates the possibility of distributing data or business logic to the client. This removes some of the application performance benefits which can be seen by a distribution strategy, however, centralized access to the business logic and business data can improve operational stability and lower costs.

A current trend is to transform mainframe based legacy systems into data- and application servers in a multi-tiered client/server or Netcentric architecture.

Overview of the Frameworks

One may ask: what frameworks one should use? This portion of the specification should help one understand:

when the various frameworks in SAF can be useful

how the frameworks are related

Frameworks Related to Delivery Vehicles

Most of the frameworks in SAF address various aspects of Delivery Vehicle architectures.

SAF provides access to the user's thought leadership and architecture frameworks for Execution, Development and Operations environments. Very briefly, SAF covers:

The Core Execution Architecture frameworks for the different architecture generations (Host, Client/Server and Netcentric). Most users will primarily use the Netcentric framework.

The Execution Architecture Extensions. This is a collection of the most common delivery vehicles that are built

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 140 of 281

for clients. These frameworks extend the core frameworks with services specific for a particular delivery vehicle.

The Development Architecture Framework. Should help one establish and operate a high-quality development environment.

The Operations Architecture Framework. Should help one establish and operate a high-quality operations environment.

To learn more about what Delivery Vehicles are, see the Delivery Vehicle Overview section. This page explains the relationships between Architecture Generations, Application Styles and Environments.

Framework Extensions and Other Frameworks

The remaining frameworks in SAF are special purpose frameworks that may not directly fit into the current Delivery Vehicle definition.

They may be extensions to the delivery vehicle frameworks such as Call Center, Mobile, eCommerce Application Framework, Middleware or Component Technologies.

Framework Recommendations

The frameworks in SAF address different aspects and areas of technology and application architecture. No single framework may cover this scope. Depending on the phase of one's project and the type of applications one's project will deliver, one may need to use different specialized frameworks.

Most implementations today may begin by considering the Netcentric Execution framework, then adding extensions for the delivery vehicles or specific technologies that your project will use. Keep in mind, however, the Development and Operations frameworks. Also, remember that some architectures will need to be built on multiple frameworks, most likely involving the Integration framework to bridge between them.

This section lists all the frameworks currently available in SAF, indicates when they may be useful, and how it relates to other frameworks:

Netcentric

When is it Useful?

This framework constitutes the core of a modem netcentric and client/server execution architecture. It will help one plan and design one's architecture by understanding what components a typical netcentric architecture should consist of.

NETCENTRIC ARCHITECTURE FRAMEWORK

FRAMEWORK OVERVIEW

Introduction

The Netcentric Architecture Framework identifies those run-time services required when an application executes in a Netcentric environment. As shown in FIG. **10**, the services can be broken down into logical areas: Presentation Services **1000**, Information Services **1002**,**1004**, Communication Services **1006**,**1008**, Communication Fabric Services **1010**, Transaction Services **1012**,**1014**, Environment Services **1016**,**1018**, Base Services **1020** and Business Logic **1022**, **1024**. This framework is an evolution of the Client Server New Age Systems Framework and is useful for technical architects involved in the selection, development and deployment of technical architectures in a Netcentric environment. More discussion of each of these logical areas is provided below. See also FIGS. **11** and **12**, which are detailed diagrams of the components of the Netcentric Architecture Framework found in FIG. **10**.

Netcentric Computing Top 10 Points

Netcentric computing represents an evolution—it builds on and extends, rather than replaces, client/server.

Netcentric computing has a greater impact on the entire business enterprise, hence greater opportunity and risk.

Definitions of Netcentric may vary. One is about reach and content.

Netcentric is not just electronic commerce; it can impact enterprises internally as well.

You can begin identifying Netcentric opportunities for clients today.

There are three basic types of Netcentric applications: advertise; inquiry; and fully interactive.

One can underestimate the impact of Netcentric on infrastructure requirements.

Build today's client/server engagements with flexibility to extend to Netcentric.

Netcentric Computing Definition:

Netcentric Computing also called Netcentric Architecture, Netcentric Technology, etc. is an emerging architecture style which expands the reach of computing both within and outside the enterprise. Netcentric enables sharing of data and content between individuals and applications. These applications provide capabilities to publish, interact or transact. Netcentric represents an evolution of Client/Server which may utilize internet technologies to connect employees, customers, and business partners.

Client/Server vs. Netcentric Computing (NCC)

NCC is a new style of computing that expands on the technological base already provided by traditional client/server systems. Many of the traditional client/server design concepts and considerations still apply to NCC.

The important differences between client/server systems and NCC systems are:

The way in which the application logic is distributed to clients is different in NCC and traditional client/server systems. In NCC systems, application logic can be packaged into components and distributed from a server machine to a client machine over a network. In traditional client/server systems, the application logic is split between the client and the server on a permanent basis; there is no dynamic distribution of application logic.

The number of tiers in NCC and traditional client/server systems is different. NCC extends the traditional two-tier client/server architecture to a n-tier architecture.

The client in NCC systems is different from a client in traditional client/server systems. The client in a NCC system is a standardized universal one; a NCC application can execute within a client that can run on multiple operating systems and hardware platforms. In traditional client/server systems, the client is custom-made for a specific operating system and hardware platform.

The way in which NCC and traditional client/server systems can be extended and adapted is different. Components enable NCC systems to be adaptable to a variety of distribution styles, from a "thin client" to a "fat client". In comparison, traditional client/server systems, once designed and built, cannot be adapted for use on more than one computing style

Tiers

Similarly to traditional client/server architectures, Netcentric architectures support a style of computing where processes on different machines communicate using mes-

sages. In this style, "client" processes delegate business functions or other tasks (such as data manipulation logic) to one or more server processes. Server processes respond to messages from clients.

Business logic can reside on both client and server. Clients are typically PCs or Workstations with a graphical user interface running in a Web browser. Servers are usually implemented on UNIX, NT or mainframe machines.

A key design decision for a client/server system is whether it should be two-tiered or multi-tiered and how business logic is distributed across the tiers. In Netcentric architectures there is a tendency to move more business logic to the server tiers, although "fatter" clients are becoming more popular with newer technologies such as Java and ActiveX.

Two-Tiered Architectures

Two-tiered architecture describes a distributed application architecture in which business applications are split into front-ends (clients) and back-ends (servers). Such a model of computing began to surface in the late 1980s and is the prominent configuration in use today by companies which have attempted to migrate to client/server based computing.

Advantages

At a minimum, a two-tiered client/server architecture assumes that an application's presentation logic resides on the client and its data management logic resides on the server. This style of computing became attractive to early adopters of client/server because it clearly addresses the inadequacies of a character-based interface. That is, it allows PC-based clients to introduce a graphical user interface (GUI) into the application environment.

Allows rapid development "out-of-the-box"

Decreased communication overhead because of a direct connection (for a small number of users)

Allows the distribution of the program's logic (application, presentation, data management)

Limitations of Two-Tiered Architecture

The use of two-tier tools has resulted in a defacto "client-heavy" or "fat-client" two-tiered model where the presentation and application logic resides on the client and data management resides on the server. In fact, the use of these tools "out-of-the-box" assumes the adoption of such a model. Unfortunately, such an architectural model falls short of addressing many important issues required of an enterprise-wide information architecture. This model of computing was actually developed for less-demanding PC environments where the database was simply a tool for decision support.

Limitations:

Limited/cost prohibitive Scalability

Limited availability

Limited reliability

Security Deficiencies

Network/Database bottlenecks

Low implementation flexibility

Limited Asynchronous processing

Three-Tiered or Multi-tiered Architectures

Three-tiered architecture describes a distributed application architecture in which business applications are separated into three logical components: presentation and control, application logic, and data management. These logical components are "clean layered" such that each runs on a different machine or platform, and communicates with the other components via a network.

A three-tiered architecture is often enhanced by the integration of distributed transaction processing middleware. This model of computing is often termed the "enhanced" client/server model. Most Netcentric architectures use a three- or four tiered approach with a web server and potentially a separate application server layer.

In the enhanced client/server model, all presentation and control logic resides on the client, all application logic resides on multiple back-end application servers, and all data management logic resides on multiple back-end database servers.

Advantages

In contrast to mainframe and two-tiered client/server computing models, the principle advantage with a three-tiered enhanced client/server architecture is that it provides the benefits of a GUI application, but also provides a level of integrity and reliability found in mainframe centralized computing. That is, it will evolve to serve high-volume, high-integrity, and high-availability environments.

Location and implementation transparency—The use of a transaction manager such as Tuxedo allows for service location independence.

Distribution of logic to optimal resource—Since the application and database functions reside on their own physical devices, each can be optimally tuned for the work they perform.

Database scaleable on throughput—In the enhanced three-tiered client/server model, client applications no longer connect directly to database servers. Instead, only application servers connect to the database servers.

Security over service resources—With the application logic residing on back-end application servers, security over the applications is made possible at various levels.

Redundancy and resiliency of services—A major disadvantage prominent in other models of computing is "single point of failure

Optimization of personnel resources—Developers can be utilized for specific talents in each tier.

Allows for asynchronous and standardized messaging—The enhanced client/server model is really a superset of the RPC-based function shipping model which provides features such as asynchronous, event-driven programming.

Administration, configuration, prioritization—The use of a transaction manager enables servers to be added, removed, or restarted dynamically. This allows for very robust, scaleable, and flexible applications.

Disadvantages

Three-tier architectures are highly flexible. This flexibility comes with the cost of being more complex to implement.

Limitations:

Additional tool (middleware) selection

Longer implementation times

Greater development costs associated with additional tier

More complex planning

Additional Skills

Extra Hardware

Greater complexity for maintenance, configuration management

PRESENTATION **1000**

Presentation Services enable an application to manage the human-computer interface. This includes capturing user actions and generating resulting events, presenting data to the user, and assisting in the management of the dialog flow of processing. FIG. **13** illustrates several components of the Presentation area of the Netcentric Architecture Framework.

Exemplary products that may be used to enable this component include Visual Basic; PowerBuilder; C++; Windows 3.x/NT/95; X-Windows/Motif; Visual C++; Borland Delphi; AC FOUNDATION for FCP.

The products listed as candidates for specific components here and below should be used with care. These examples do not provide an all-inclusive list, nor do they necessarily represent the current market leaders. They are there to provide an example of products that may enable the component services.

Window System **1300**

Typically part of the operating system, the Window System Services provide the base functionality for creating and managing a graphical user interface (GUI)—detecting user actions, managing windows on the display, and displaying information in windows.

Implementation Considerations

Windowing systems expose their functionality to application programs through a set of application programming interfaces (APIs). For the Microsoft windowing platform, this API is called Win32. The Win32 API is a documented set of over 500 C functions that allow developers to access the functionality of the windowing system as well as various other operating system functions. While it is possible for developers to directly call the Win32 API or its equivalent on other platforms using a C language compiler, most business application development is done using higher level development languages such as Visual Basic or PowerBuilder which make the lower level calls to the operating systems on behalf of the developer.

Exemplary products that may be used to enable this component include Microsoft Windows; Windows 95; Windows NT; Macintosh OS; Program Manager for OS/2; X-Windows/Motif; JavaOS

Desktop Manger **502**

Desktop Manager Services implement the desktop metaphor. The desktop metaphor as the name suggests is a style of user interface that tries to emulate the idea of a physical desktop allowing you to place documents on the desktop, launch applications by clicking on a graphical icon, or discard files by dragging them onto a picture of a waste basket. Most Window Systems contain elementary Desktop Manager functionality (e.g., the Windows 95 desktop), but often more user friendly or functional Desktop Manager Services are required.

Microsoft Windows 95 task bar; Norton Navigator; Xerox Tabworks; Starfish Software Dashboard

Product Considerations

Exemplary products that may be used to enable this component include:

- Microsoft Windows 95 task bar—provides a launch bar which allows users to access recently used documents, launch applications, or switch between active applications. The Windows 95 desktop and launch bar are programmable allowing users to extend and customize the desktop manager for their specific application. For example, the desktop can be extended with icons or Start Menu options for creating a new customer account or finding an order.
- Norton Navigator—provides multiple virtual desktops, enhanced file management including direct FTP connectivity, long file name support for some 16-bit applications, file un-erase, and other features; targeted at users who often interact with the Windows 95 desktop.
- Xerox Tabworks—presents the user with a notebook metaphor for application and document access; allows creation of tabbed sections which contain related files (e.g., Winston Account or New Product Launch) for easier access.
- Starfish Software Dashboard—a desktop utility designed to simplify application and system management; provides quick launch buttons, system resource gauge, drag-and-drop printing and faxing, calendar, etc.

Form **1304**

Form Services enable applications to use fields to display and collect data. A field may be a traditional 3270-style field used to display or input textual data, or it may be a graphical field such as a check box, a list box or an image. Form Services provide support for:

- Display—support the display of various data types (e.g., text, numeric, date, etc.) in various formats (e.g., American/European date, double-byte characters, icons, etc.)
- Input/Validation—enable applications to collect information from the user, edit it according to the display options, and perform basic validation such as range or format checks.
- Mapping Support—eliminate the need for applications to communicate directly with the windowing system; rather, applications retrieve or display data by automatically copying the contents of a window's fields to a copybook structure in memory. These Services may also be used to automate the merging of application data with pre-defined electronic form templates.
- Field Interaction Management—coordinate activity across fields in a window by managing field interdependencies and invoking application logic based on the state of fields and user actions. For example, the Field Interaction Manager may disable the "OK" button until all required input fields contain valid data. These services significantly reduce the application logic complexity inherent to an interactive windowed interface.

Implementation Considerations

In traditional client/server applications, Forms are windows that contain widgets (text fields, combo-boxes, etc.) and business logic. Form development tools such as Visual Basic, PowerBuilder, etc. allow the Form designer to specify page layout, entry fields, business logic, and routing of forms. From a developers perspective, these products typically expose Form and control handling functionality as a set of proprietary or product specific APIs.

In addition to the traditional tools (e.g., Visual C++, Visual Basic, PowerBuilder), Netcentric technologies have introduced new tools that can be used to develop Forms. For example, a developer can use Symantec Visual Café to create a Java application that will execute directly on the users desktop without any interaction with a browser.

Today most Netcentric applications are Web based and are launched from the Web browser. Additionally, one is now beginning to see other types of Netcentric solutions. For example, PointCast is a Netcentric application located on the users machine; it relies on the Internet to deliver stock prices, news headings, sports updates, etc. to the user. However, it is not launched from the Web browser—it is its own application. In the future there will be more Netcentric applications that use this approach for delivering information.

Product Considerations

What Level of Technical Support, Documentation, and Training is Required to Ensure the Productivity of Developers?

The extent of support (on-site, phone, bulletin board, world-wide, etc.), quality of documentation, and availability and location of education/training should be considered.

What Functions are Required in the Control set?

At the minimum a tool should support basic widgets (push buttons, list boxes, etc.), window styles, (multi-window, multi-document, paned-window), and menu styles, along with validation and inter-application communication. Consideration should also be given as to the extensibility of the toolset via add-ons and third party products.

Can the Tool be Used for Both Prototyping and GUI Design?

The ability to use a single tool for both prototyping and GUI design will reduce the development learning curve. One should also consider how well the tool integrates will all other development tools.

What Platform(s) are Supported?

The platform(s) that must be supported, i.e., MS-DOS, Windows, IBM OS/2, UNIX, or UNIX Motif, is an important consideration, as are any hardware restrictions.

What Type of Learning Curve is Associated With the Tool?

Developers using the product should be able to become productive quickly. Factors which reduce the learning curve include an easy to learn and intuitive interface, thorough and clear documentation, and on-line help.

If the Tool is Also Going to be Used for Application Development, How Well Does the Tool Perform During Production?

Computational, network, data retrieval, and display speeds differ for products. Factors to consider are whether the application will consist of heavy data entry, transaction processing, or a large user base.

How Much Does the Tool Cost?

Product components, maintenance agreements, upgrades, run-time licenses, and add-on packages should be considered.

Does the Product Integrate With Other Tools and/or Support Other Tools in the Development and Execution Environments?

It is important to determine how well the product integrates with other design and development tools, presentation services (graphics, multi-media, etc.), data access services (databases and database API libraries), distribution services (distributed TP monitor), transmission services (SNA, HLLAPI, etc.), data dictionary, desktop applications, and programming languages for call-out/call-in. Additional consideration should be given to add-on and third-party products/enhancements such as specialized widgets, report writers and case tools.

Will the Tool be Used With a Large Development Team?

If the development team is more than 5 people, a tool should provide support for multiple developers. This support includes features such as object check-in/check-out, a central design repository for the storage of application objects and user interface definitions, and version control. Additionally, the development team should be able to cleanly divide the application(s) into pieces which can be worked on by multiple people.

What Protocols are Used to Communicated With the Database?

Important considerations include the supported databases and protocols used to communicated with the databases. The tool must support the selected database. Additionally, if the database selection may change, it is important that the tool have the ability to support other databases with minimal impact on the application development. Native database interfaces tend to have better performance than open standards such as ODBC.

Will the Design Tool be Used for Programming of Client Applications? What Programming Language is Supported?

If the design tool is used for programming, there are several features of a tool which must be considered. These features can have an impact on the productivity of programmers, performance of the applications, skill sets required, and other tools required for development. These features include:

What programming language is supported? Is the programming language interpretive or compiled? Is it object oriented or structured procedural language?

Does the tool support programming extensions to Dynamic Link Libraries?

What are the debugging capabilities of the tool?

Is the Tool Scalable?

The tool should be scalable to support growth in application size, users, and developers.

Exemplary products that may be used to implement this component include JetForms JetForm Design; Lotus Forms; Visual Basic.

JetForms JetForm Design—provides tools to design, fill, route, print and manage electronic forms, helping organizations reduce costs and increase efficiency by automating processing of forms across local and wide area networks as well as the Internet. Lotus Forms—Lotus Development Corporations electronic forms software provides tools to design, route and track forms to automate business processes for the workgroup or the extended enterprise. Lotus Forms is designed to run with Lotus Notes or as a standalone application. It is comprised of two parts: Forms Designer, an application-development version, and Forms Filler, a runtime version for users. Visual Basic—a development tool that provides a comprehensive development environment for building complex applications.

User Navigation **1306**

User Navigation Services provide a user with a way to access or navigate between functions within or across applications. Historically, this has been the role of a text-based menuing system that provides a list of applications or activities for the user to choose from.

Client/server technologies introduced new navigation metaphors. A method for allowing a user to navigate within an application is to list available functions or information by means of a menu bar with associated pull-down menus or context-sensitive pop-up menus. This method conserves screen real-estate by hiding functions and options within menus, but for this very reason can be more difficult for first time or infrequent users. This point is important when implementing electronic commerce solutions where the target customer may use the application only once or very infrequently (e.g., purchasing auto insurance).

Additionally, client/server development tools such as Visual Basic and PowerBuilder do not,provide specific services for graphical navigation, but the effect can be recreated by selecting (i.e., clicking on) graphical controls, such as picture controls or iconic push-buttons, programmed to launch a particular window.

A major advantage of the graphical user interface is the fact that it allows multiple windows to be open at one time.

Implementation Considerations

Is There a Need to Manage Multiple Instances of a Window Object?

Windows Interaction Manager provides the application with facilities to open multiple instances of the same window. This component provides an option parameter that will let the application developers enable or disable the ability to

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 144 of 281

open the same window with the same key data (that is, a duplicate instance).

Do you Need to Pass Messages Between Windows?

Windows Interaction Manager provides the facility to pass messages between windows within one application. This allows one window to trigger an event/action on another related window.

Do Multiple Applications Need to Pass Messages Between Each Other?

Windows Interaction Manager provides the facility to pass messages between windows from different applications residing on the same machine. This allows one window to trigger an event/action on an related window when certain actions (user or environment) occur.

If information needs to be shared between applications on different machines, Window Interaction Management cannot be used. This type of data sharing requires a special architecture component called Communication, which is more network orientated.

Is There a Need for Object Registration/de-registration?

Windows Interaction management allows the application to control and manage the opening and closing of multiple windows by—maintaining the parent-child relationship, controlling multiple instances of similar windows, maintaining key data-window relationship. This allows the user to work in a controlled and, well managed, environment.

Web Browser **1308**

Web Browser Services allow users to view and interact with applications and documents made up of varying data types, such as text, graphics, and audio. These services also provide support for navigation within and across documents no matter where they are located, through the use of links embedded into the document content. Web Browser Services retain the link connection, i.e., document physical location, and mask the complexities of that connection from the user. Web Browser services can be further subdivided into: Browser Extension, Form, and User Navigation.

Parlez-vous Internet?

The Elements of Web Style

Language philosopher Benjamin Whorf once said, "We dissect nature along lines laid down by our native language. Language is not simply a reporting device for experience, but a defining framework for it." This notion is especially true when applied to the World Wide Web. The evolution of the Web from a rigid, text-centric village to an elastic, multimedia-rich universe has been driven by modifications to the languages behind it. The Internet is at a crucial point in its development as a number of enhancements for extending Web technology come under scrutiny by Internet standards groups. These enhancements will ultimately push the Web into the realms of distributed document processing and interactive multimedia.

SGML: in the Beginning . . .

Although the World Wide Web was not created until the early 1990s, the language behind it dates back to the genesis of the Internet in the 1960s. Scientists at IBM were working on a Generalized Markup Language (GML) for describing, formatting, and sharing electronic documents. Markup refers to the practice in traditional publishing of annotating manuscripts with layout instructions for the typesetters.

In 1986, the International Standards Organization (ISO) adopted a version of that early GML called Standard Generalized Markup Language (SGML). SGML is a large and highly-sophisticated system for tagging documents to ensure that their appearance will remain the same regardless of the type of platform used to view them. Designers use SGML to create Document Type Definitions (DTDs), which detail how tags (also known as format codes) are defined and interpreted within specified documents. These tags can be used to control the positioning and formatting of a document's text and images. SGML is used for large, complex, and highly-structured documents that are subject to frequent revisions, such as dictionaries, indexes, computer manuals, and corporate telephone directories.

HTML: SGML for Dummies?

While creating the World Wide Web in the early 1990s, scientists at CERN discovered that in spite of its power and versatility, SGML's sophistication did not allow for quick and easy Web publishing. As a result, they developed HyperText Markup Language (HTML), a relatively simple application of SGML. This simplicity has contributed to the exponential growth of the Web over the last few years. HTML files are written in plain text and can be created using any text editor from the most robust Web page authoring software (such as Microsoft's FrontPage or Sausage Software's HotDog) to the anemic Notepad utility included with Microsoft's Windows operating system.

As with many languages, HTML is in a state of constant evolution. The World Wide Web Consortium W3C oversees new extensions of HTML developed by both software companies (such as Microsoft and Netscape Communications) and individual Web page authors and ensures that each new specification is fully-compatible with previous ones. Basic features supported by HTML include headings, lists, paragraphs, tables, electronic forms, in-line images (images next to text), and hypertext links. Enhancements to the original HTML 1.0 specification include banners, the applet tag to support Java, image maps, and text flow around images.

The W3C also approved the specification for version 4.0 of HTML (http://www.w3.org/TR/REC-html40). This specification builds upon earlier iterations of HTML by enabling Web authors to include advanced forms, in-line frames, and enhanced tables in Web pages. HTML 4.0 also allows authors to publish pages in any language, and to better manage differences in language, text direction, and character encoding.

Perhaps most significantly, HTML 4.0 increases authors' control over how pages are organized by adding support for Cascading Style Sheets CSS Style sheets contain directions for how and where layout elements such as margins, fonts, headers, and links are displayed in Web pages. With CSS, authors can use programming scripts and objects to apply multiple style sheets to Web pages to create dynamic content. CSS can also be used to centralize control of layout attributes for multiple pages within a Web site, thus avoiding the tedious process of changing each page individually.

Dynamic HTML: Dyn-o-mite!

HTML's simplicity soon began to limit authors who demanded more advanced multimedia and page design capabilities. Enter Dynamic HTML DHTML As an extension of HTML, DHTML allows Web pages to function more like interactive CD-ROMs by responding to user-generated events. DHTML allows Web page objects to be manipulated after they have been loaded into a browser. This enables users to shun plug-ins and Java applets and avoid bandwidth-consuming return trips to the server. For example, tables can expand or headers can scurry across the page based on a user's mouse movements.

Unfortunately, the tremendous potential offered by DHTML is marred by incompatible standards. At the heart of the DHTML debate is a specification called the Document Object Model DOM The DOM categorizes Web page elements—including text, images, and links—as objects and specifies the attributes that are associated with each object. The DOM makes Web document objects accessible to

scripting languages such as JavaScript and VisualBasic Script (VBScript), which can be used to change the appearance, location, and even the content of those objects in real-time.

Microsoft's Internet Explorer 4.0 supports a W3C "Working Draft" DOM specification that uses the CSS standard for layout control and Web document object manipulation. In contrast, Netscape's implementation of DHTML in Communicator 4.0 uses a proprietary "Dynamic Layers" tag, which assigns multiple layers to a page within which objects are manipulated. As a result, Web pages authored using either version of DHTML may not be viewed properly using the other's browser. XML: X marks the spot

HTML 4.0 and Dynamic HTML have given Web authors more control over the ways in which a Web page is displayed. But they have done little to address a growing problem in the developer community: how to access and manage data in Web documents so as to gain more control over document structure. To this end, leading Internet developers devised Extensible Markup Language (XML), a watered-down version of SGML that reduces its complexity while maintaining its flexibility. Like SGML, XML is a meta-language that allows authors to create their own customized tags to identify different types of data on their Web pages. In addition to improving document structure, these tags will make it possible to more effectively index and search for information in databases and on the Web.

XML documents consist of two parts. The first is the document itself, which contains XML tags for identifying data elements and resembles an HTML document. The second part is a DTD that defines the document structure by explaining what the tags mean and how they should be interpreted. In order to view XML documents, Web browsers and search engines will need special XML processors called "parsers." Currently, Microsoft's Internet Explorer 4.0 contains two XML parsers: a high-performance parser written in C++ and another one written in Java.

A number of vendors plan to use XML as the underlying language for new Web standards and applications. Microsoft uses XML for its Channel Definition Format, a Web-based "push" content delivery system included in Internet Explorer 4.0. Netscape will use XML in its Meta Content Framework to describe and store metadata, or collections of information, in forthcoming versions of Communicator. XML is currently playing an important role the realm of electronic commerce via the Open Financial Exchange, an application developed by Microsoft, Intuit, and CheckFree for conducting electronic financial transactions. Similarly, HL7, a healthcare information systems standards organization, is using XML to support electronic data interchange EDI of clinical, financial, and administrative information (http://www.mcis.duke.edu/standards/HL7/sigs/sgml/index.html).

Meet Cousin VRML

In 1994, a number of Internet thought leaders, including Tim Berners-Lee—the "father" of the Web—met to determine how they could bring the hot, new technology known as virtual reality VR to the Web. VR refers to the use of computers to create artificial and navigable 3-D worlds where users can create and manipulate virtual objects in real time. This led to the creation of Virtual Reality Modeling Language (VRML—pronounced "ver-mul"). VRML is technically not a markup language because it uses graphical rather than text-based file formats.

In order to create 3-D worlds and objects with VRML, users need a VRML editor such as Silicon Graphics' Cosmo Worlds (http://cosmo.sgi.com/products/studio/worlds). To view VRML content, users need either a VRML browser or a VRML plug-in for standard HTML browsers. Leading VRML plug-ins include Cosmo Player from Silicon Graphics (http://vrml.sgi.com/cosmoplayer), Liquid Reality from Microsoft's DimensionX subsidiary (http://www.microsoft.com/dimensionx), OZ Virtual from OZ Interactive (http://www.oz.com/ov/main_bot.html), and WorldView from Intervista (http://www.intervista.com/products/worldview/index.html), These plug-ins can typically be downloaded for free from the Web.

VRML is capable of displaying static and animated objects and supports hyperlinks to multimedia formats such as audio clips, video files, and graphical images. As users maneuver through VRML worlds, the landscape shifts to match their movements and give the impression that they are moving through real space. The new VRML 2.0 specification finalized in August 1996 intensifies the immersive experience of VR worlds on the Web by enabling users to interact both with each other and with their surroundings. Other new features supported by VRML 2.0 include richer geometry description, background textures, sound and video, multilingual text, Java applets, and scripting using VBScript and JavaScript. VRML will become a significant technology in creating next-generation Internet application as the language continues to mature and its availability increases.

The Future: Give us a Big SMIL

The Web has come a long way since the codification of HTML 1.0. It has moved from simple text-based documents that included headings, bulleted lists, and hyperlinks to dynamic pages that support rich graphic images and virtual reality. So what next for the Web? The answer resides in a Synchronized Multimedia Integration Language (SMIL), a new markup language being developed by the W3C. SMIL will allow Web authors to deliver television-like content over the Web using less bandwidth and a simple text editor, rather than intricate scripting. SMIL is based on XML and does not represent a specific media format. Instead, SMIL defines the tags that link different media types together. The language enables Web authors to sort multimedia content into separate audio, video, text, and image files and streams which are sent to a user's browser. The SMIL tags then specify the "schedule" for displaying those components by determining whether they should be played together or sequentially. This enables elaborate multimedia presentations to be created out of smaller, less bandwidth-consuming components.

Implementation Considerations

Many features such as graphics, frames, etc. supported by Web Browsers today were not available in initial releases. Furthermore, with every new release the functionality supported by Web Browsers keeps growing at a remarkable pace.

Much of the appeal of Web Browsers is the ability to provide a universal client that will offer users a consistent and familiar user interface from which many types of applications can be executed and many types of documents can be viewed, on many types of operating systems and machines, as well as independent of where these applications and documents reside.

Web Browsers employ standard protocols such as Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP) to provide seamless access to documents across machine and network boundaries.

The distinction between the desktop and the Web Browser narrowed with the release of Microsoft IE 4.0, which integrated Web browsing into the desktop, and gave a user the ability to view directories as though they were Web

pages. Web Browser, as a distinct entity, may even fade away with time.

Exemplary products that may be used to implement this component includes Netscape Navigator; Netscape Communicator; Microsoft Internet Explorer; Netscape LiveWire; Netscape LiveWire Pro; Symantec Visual Cafe; Microsoft Front Page; Microsoft Visual J++; IBM VisualAge.

Execution Products:

Netscape Navigator or Communicator—one of the original Web Browsers, Navigator currently has the largest market share of the installed browser market and strong developer support. Communicator is the newest version with add-on collaborative functionality

Microsoft Internet Explorer (IE)—a Web Browser that is tightly integrated with Windows and supports the major features of the Netscape Navigator as well as Microsoft's own ActiveX technologies.

Development Products:

Web Browsers require new or at least revised development tools for working with new languages and standards such as HTML, ActiveX and Java. Many browser content development tools are available. The following are several representative products:

Netscape LiveWire and LiveWire Pro—visual tool suite designed for building and managing complex, dynamic Web sites and creating live online applications.

Symantec Visual Café—the first complete Rapid Application Development (RAD) environment for Java; it allows developers to assemble complete Java applets and applications from a library of standard and third party objects. Visual Café also provides an extensive set of text based development tools.

Microsoft FrontPage—Web site management tool that supports web page creation, web site creation, page and link management and site administration.

Microsoft Visual J++—a product similar to Visual C++, VJ++ allows the construction of Java and ActiveX applications through an integrated graphical development environment.

IBM VisualAge for Java—a product similar to VisualAge for Smalltalk, VJ++ allows the construction of Java applications through an integrated graphical development environment. It supports JavaBeans. Used by Eagle team for the Eagle JavaBeans reference application

Browser Extension **1310**

Browser Extension Services provide support for executing different types of applications from within a Browser. These applications provide functionality that extend Browser capabilities. The key Browser Extensions are:

Plug-in—a term coined by Netscape, a plug-in is a software program that is specifically written to be executed within a browser for the purpose of providing additional functionality that is not natively supported by the browser, such as viewing and playing unique data or media types. Typically, to use a plug-in, a user is required to download and install the Plug-in on his/her client machine. Once the Plug-in is installed it is integrated into the Web browser. The next time a browser opens a Web page that requires that Plug-in to view a specific data format, the browser initiates the execution of the Plug-in. Until recently Plug-ins were only accessible from the Netscape browser. Now, other browsers such as Microsoft's Internet Explorer are beginning to support Plug-in technology as well. Also, Plug-ins written for one browser will generally need to be modified to work with other browsers. Plug-ins are also operating system dependent. Therefore, separate versions of a Plug-in may be required to support Windows, Macintosh, and Unix platforms.

Helper Application/Viewer—is a software program that is launched from a browser for the purpose of providing additional functionality to the browser. The key differences between a helper application or sometimes called a viewer and a plug-in are:

* How the program is integrated with the Web browser—unlike a plug-in, a helper application is not integrated with the Web Browser, although it is launched from a Web browser. A helper application generally runs in its own window, contrary to a plug-in which is generally integrated into a Web page.
* How the program is installed—like a plug-in, the user installs the helper application. However, because the helper application is not integrated with the browser, the user tends to do more work during installation specifying additional information needed by the browser to launch the helper application.
* How the program is initiated—the user tends to initiate the launching of the helper application, unlike a plug-in where the browser does the initiation.
* From where the program is executed—the same helper application can be executed from a variety of browsers without any updates to the program, unlike a plug-in which generally needs to be updated for specific browsers. However, helper applications are still operating system dependent.

Java applet—a program written in Java that runs within or is launched from the client's browser. This program is loaded into the client device's memory at runtime and then unloaded when the application shuts down. A Java applet can be as simple as a cool animated object on an HTML page, or can be as complex as a complete windows application running within the browser.

ActiveX control—is also a program that can be run within a browser, from an application independent of a browser, or on its own. ActiveX controls are developed using Microsoft standards that define how re-usable software components should be built. Within the context of a browser, ActiveX controls add functionality to Web pages. These controls can be written to add new features like dynamic charts, animation or audio.

Implementation Considerations

Viewers and plug-ins are some of the most dynamic segments of the browser market due to quickly changing technologies and companies. What was yesterday a plug-in or a viewer add-on often becomes a built-in capability of the browser in its next release.

Exemplary products that may be used to implement this component include Real Audio Player; VDOLive; Macromedia Shockwave; Internet Phone; Web 3270.

Real Audio Player—a plug-in designed to play audio and video in real-time on the Internet without requiring to download the entire audio file before you can begin listening, or a video file before you can begin viewing.

Macromedia Shockwave—a plug-in used to play back complex multimedia documents created using Macromedia Director or other products.

Internet Phone—one of several applications which allow two-way voice conversation over the Internet, similar to a telephone call.

Web3270—a plug-in from Information Builders that allows mainframe **3270**-based applications to be

viewed across the Internet from within a browser. The Web3270 server provides translation services to transform a standard 3270 screen into an HTML-based form. Interest in Web3270 and similar plug-ins has increased with the Internets ability to provide customers and trading partners direct access to an organizations applications and data. Screen scraping plug-ins can bring legacy applications to the Internet or intranet very quickly.

Form **1312**

Like Form Services outside the Web Browser, Form Services within the Web Browser enable applications to use fields to display and collect data. The only difference is the technology used to develop the Forms. The most common type of Forms within a browser are Hypertext Markup Language (HTML) Forms. The HTML standard includes tags for informing a compliant browser that the bracketed information is to be displayed as an editable field, a radio button, or other form-type control. Currently, HTML browsers support only the most rudimentary forms—basically providing the presentation and collection of data without validation or mapping support. When implementing Forms with HTML, additional services may be required such as client side scripting (e.g., VB Script, JavaScript).

Additionally Microsoft has introduced ActiveX documents which allow Forms such as Word documents, Excel spreadsheets, Visual Basic windows to be viewed directly from Internet Explorer just like HTML pages.

Different technologies may be used to create Forms that are accessible outside of the browser from those that are accessible within the browser. However, with the introduction of ActiveX documents these differences are getting narrower.

Exemplary products that may be used to implement this component include JetForms JetForm Design; Lotus Forms; Visual Basic; Front Page.

FrontPage—web site management tool that supports web page creation, web site creation, page and link management and site administration.

User Navigation **1314**

Like User Navigation Services outside the Web Browser, User Navigation Services within the Web Browser provide a user with a way to access or navigate between functions within or across applications. These User Navigation Services can be subdivided into three categories:

Hyperlink—the Internet has popularized the use of underlined key words, icons and pictures that act as links to further pages. The hyperlink mechanism is not constrained to a menu, but can be used anywhere within a page or document to provide the user with navigation options. It can take a user to another location within the same document or a different document altogether, or even a different server or company for that matter. There are three types of hyperlinks:

Hypertext is very similar to the concept of Context Sensitive Help in Windows, where the reader can move from one topic to another by selecting a highlighted word or phrase.

Icon is similar to the hypertext menu above, but selections are represented as a series of icons. The HTML standard and popular browsers provide hyperlinking services for non-text items such as graphics.

Image Map is also similar to the hypertext menu above, but selections are represented as a series of pictures. A further evolution of the image map menu is to display an image depicting some place or thing (e.g., a picture of a bank branch with tellers and loan officers).

Custoniized Menu—a menu bar with associated pull-down menus or context-sensitive pop-up menus. However, as mentioned earlier this method hides functions and options within menus and is difficult for infrequent users. Therefore, it is rarely used directly in HTML pages, Java applets or ActiveX controls. However, this capability might be more applicable for intranet environments where the browsers themselves need to be customized (e.g., adding custom pull-down menus within Internet Explorer) for the organizations specific business applications.

Virtual Reality—A virtual reality or a virtual environment interface takes the idea of an image map to the next level by creating a 3-dimensional (3-D) environment for the user to walk around in. Popularized by PC games like Doom, the virtual environment interface can be used for business applications. Imagine walking through a shopping mall and into and around virtual stores, or flying around a 3-D virtual resort complex you are considering for a holiday.

To create sophisticated user navigation interfaces such as these requires additional architectural services and languages. The Virtual Reality Modeling Language (VRML) is one such language gaining in popularity.

Implementation Considerations

The hyperlink metaphor makes it possible for the user to jump from topic to topic instead of reading the document from beginning to end. For many types of applications, this can create a more user-friendly interface, enabling the user to find information faster.

An image map menu can be useful where all users share some visual model for how business is conducted, and can be very engaging, but also painfully slow if even a moderate speed communications connection is required. Additional Image Map Services are required to map the location of user mouse clicks within the image to the corresponding page or window which is to be launched.

Exemplary products that may be used to implement this component include Silicon Graphics Open Inventor; VREAM VRCreator; DimensionX Liquid Reality.

There are many toolkits and code libraries available to speed development of applications utilizing Reality services. Below are some representative products:

Silicon Graphics Open Inventor—an object-oriented 3-D toolkit used to build interactive 3-D graphics using objects such as cameras, lights and 3-D viewers; provides a simple event model and animation engine.

VREAM VRCreator—a toolkit for building interactive virtual reality environments; supports gravity, elasticity, and throw-ability of objects, textured and colored 3-D objects and construction of networked multi-participant worlds. Provides support for ActiveX.

DimensionX Liquid Reality—VRML 2.0 platform written in Java, which provides both a viewer for viewing VRML content and a toolkit of Java classes for creating powerful 3-D applications. It supports more than 250 classes for 3-D content creation.

Report and Print **1316**

Report and Print Services support the creation and on-screen previewing of paper or photographic documents which contain screen data, application data, graphics or images.

Implementation Considerations

Printing services must take into consideration varying print scenarios common in Netcentric environments, including: varying graphics/file types (Adobe.PDF, .GIF, .JPEG), page margins and breaks, HTML constructs including tables and frames, headers/titles, extended character set support, etc.

Is There a Need for Reporting or Decision Support?

Use report writers when you need to transform user data into columnar reports, forms, or mailing lists that may require sophisticated sorting and formatting facilities. This generally occurs for two reasons. The first is building "production reports" (i.e., reports that are built once and then used repeatedly, generally on a daily/weekly/monthly basis). The second is ad hoc reporting and decision support. Products targeted at one or the other use will have different facilities. (source is market research)

Is There a Need to Ease Access to Corporate Data?

Use report writers when users require easy and quick access to corporate data. Since developers can deliver reports as run-time applications, users are shielded from having to learn complicated databases in order to access information. All a user has to do to retrieve the data is click on an icon to launch a report. Because these run-time applications are smaller than normal applications, they launch faster and require very little training to operate. (source is market research)

Product Considerations

Buy vs. Build

There are numerous packaged controls on the market today that support basic report and print capability. However, a careful evaluation of both functions and features and vendor viability must be completed before a decision can be made. Architects must additionally be sure to evaluate that controls will support all required environments, are small in size and extensible as requirements demand.

How Important is Performance?

In general, performance of data access and printing should be considered. Some typical benchmark tests include table scan, single-table report, joined table report, and mailing label generation times. (source is market research)

What is the Budget?

Per developer costs as well as run time licensing fees, maintenance costs, support fees, and upgrade charges should be considered.

Do I Have Another Component That Satisfies This Requirement?

Many databases and application development tools are shipped with built in or add-on report writing capability. However, stand-alone report writers: (1) are more powerful and flexible, especially when dealing with multiple data sources and a wide variety of fornats; (2) can retrieve information from more data sources than the bundled report writers and can create reports from several data sources simultaneously; (3) excel in ease of use, both in designing and generating reports; (4) offer better tools and more predefined reports; and (5) have faster engines. (source is market research)

Does the Product Integrate With the Existing or Proposed Architecture?

It is important to consider how well a product integrates with desktop tools (word processing, spreadsheet, graphics etc.) and application development programs. These items can be used to extend the capabilities of the reporting package.

What Databases Does the Product Support?

A product should support the most widely used PC file formats and Client/Server databases. It may be necessary to consider the type of support. For example, native database interfaces tend to have better performance than open standards such as ODBC. Another possible consideration is how well the product accesses multiple files or databases. (source is market research)

What are the Required Features of the Tool?

Features to look for include but are not limited to:

WYSIWYG print preview

Ability to create views—prevents users from getting overwhelmed with choices when, selecting a table, acts as a security system by controlling which users have access to certain data, and increases performance since only the data users need gets downloaded to the report engine, thereby reducing network traffic.

Data dictionary—store predefined views, formats, and table and field name aliases

User friendly query tool

Scripting or macro language

Supported data types and formats

Formatting capabilities (page orientation, fonts, colors, margins, condensed printing, etc.)

Supported report types

Aggregate functions.

Is the Intention to Create Production Reports or Facilitate End User Queries?

Ease of use will be of major importance for end user query and decision support type applications. In contrast, functionality that allows for the implementation of complex reporting requirements will outweigh ease of use for applications whose objective is creating production reports.

Direct Manipulation **1318**

Direct Manipulation Services enable applications to provide a direct manipulation interface (often called "drag & drop"). A direct manipulation interface allows users to manage multiple "application objects" by manipulating visual representations of those objects. For example, a user may sell stock by dragging "stock" icons out of a "portfolio" icon and onto a "trading floor" icon. Direct Manipulation Services can be further divided as' follows:

Display: These services enable applications to represent application objects as icons and control the display characteristics (color, location, etc.) of these icons.

Input/Validation: These services enable applications to invoke validation or processing logic when an end user "acts on" an application object. "Acting on" an object may include single clicking, double clicking, dragging, or sizing.

Input Device **1320**

Detect user input from a variety of input technologies (i.e. pen based, voice recognition, touch-screen, mouse, digital camera, etc.).

Implementation Considerations

Voice response systems are used to provide prompts and responses to users through the use of phones. Voice response systems have scripted call flows which guide a caller through a series of questions. Based on the users key pad response, the voice response system can execute simple calculations, make database calls, call a mainframe legacy application or call out to a custom C routine. Leading voice response system vendors include VoiceTek and Periphonics.

Voice recognition systems are becoming more popular in conjunction with voice response systems. Users are able to speak into the phone in addition to using a keypad. Voice recognition can be extremely powerful technology in cases where a key pad entry would be limiting (e.g., date/time or location). Sophisticated voice recognition systems have been built which support speaker-independence, continuous speech and large vocabularies.

INFORMATION **1002,1004**

FIG. **14** illustrates several components of the Information Services of the present invention. Information Services

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 149 of 281

manage electronic data assets and enable applications to access and manipulate data stored locally or remotely in documents or databases. They minimize an application's dependence on the physical storage and location within the network. Information Services can be grouped into two categories: Database Services, and Document Services

Database Services **1402**

Database Services are responsible for providing access to a local or a remote database, maintaining integrity of the data within the database and supporting the ability to store data on either a single physical platform, or in some cases across multiple platforms. These services are typically provided by DBMS vendors and accessed via embedded or call-level SQL variants and supersets. Depending upon the underlying storage model, non-SQL access methods may be used instead.

Many of the Netcentric applications are broadcast-type applications, designed to market products and/or publish policies and procedures. Furthermore, there is now a growth of Netcentric applications that are transaction-type applications used to process a customers sales order, maintenance request, etc. Typically these type of applications require integration with a database manager. Database Services include:

'Storage Services, Indexing Services, Security Services, Access Services, and Replication/Synchronization Services

Implementation Considerations

The core database services such as Security, Storage and Access are provided by all major RDBMS products, whereas the additional services of Synchronization and Replication are available only in specific products.

Product Considerations

Oracle 7.3; Sybase SQL Server; Informix; IBM DB/2; Microsoft SQL Server

Oracle 7.3—market leader in the Unix client/server RDBMS market, Oracle is available for a wide variety of hardware platforms including MPP machines. Oracles market position and breadth of platform support has made it the RDBMS of choice for variety of financial, accounting, human resources, and manufacturing application software packages. Informix—second in RDBMS market share after Oracle, Informix is often selected for its ability to support both large centralized databases and distributed environments with a single RDBMS product. Sybase SQL Server—third in RDBMS market share, Sybase traditionally focused upon medium-sized databases and distributed environments; it has strong architecture support for database replication and distributed transaction processing across remote sites.

IBM DB2—the leader in MVS mainframe database management, IBM DB2 family of relational database products are designed to offer open, industrial strength database management for decision support, transaction processing and line of business applications. The DB2 family now spans not only IBM platforms like personal computers, AS/400 systems, RISC System/6000 hardware and IBM mainframe computers, but also non-IBM machines such as Hewlett-Packard and Sun Microsystems. Microsoft SQL Server—the latest version of a high-performance client/server relational database management system. Building on version 6.0, SQL Server 6.5 introduces key new features such as transparent distributed transactions, simplified administration, OLE-based programming interfaces, improved support for industry standards and Internet integration.

Replication/Synchronization **1404**

Replication Services support an environment in which multiple copies of databases must be maintained. For example, if ad hoc reporting queries or data warehousing applications can work with a replica of the transaction database, these resource intensive applications will not interfere with mission critical transaction processing. Replication can be either complete or partial. During complete replication all records are copied from one destination to another, while during partial replication, only a subset of data is copied, as specified by the user or the program. Replication can also be done either real-time or on-demand (i.e., initiated by a user, program or a scheduler). The following might be possible if databases are replicated on alternate server(s): better availability or recoverability of distributed applications; better performance and reduced network cost, particularly in environments where users are widely geographically dispersed; etc.

Synchronization Services perform the transactions required to make one or more information sources that are intended to mirror each other consistent. This function may especially valuable when implementing applications for users of mobile devices because it allows a working copy of data or documents to be available locally without a constant network attachment. The emergence of applications that allow teams to collaborate and share knowledge has heightened the need for Synchronization Services in the execution architecture.

The terms Replication and Synchronization are used interchangeably, depending on the vendor, article, book, etc. For example, when Lotus Notes refers to Replication, it means both a combination of Replication and Synchronization Services described above. When Sybase refers to Replication it only means copying data from one source to another.

Implementation Consideration

Replication/Synchronization Services are sometimes supplied as part of commercial databases, document management systems or groupware products such as Lotus Notes, Microsoft Exchange, Oracle, etc.

With Windows 95 and Windows NT 4.0, Microsoft has also introduced the concept of Replication/Synchronization Services into the operating system. Through the briefcase application users can automatically synchronize files and SQL data between their Windows PC and a Windows NT server. Underlying this application is the user-extensible Win32 synchronization services API which can be used to build custom synchronization tools.

Are Changes in Data Usage Anticipated?

Data can be dynamically changed to accommodate changes in how the data is used.

Is it Desirable to Shield the User From the Data Access Process?

A replicated database often consolidates data from heterogeneous data sources, thus shielding the user from the processes required to locate, access and query the data.

What are the Availability Requirements of the System?

Replication provides high availability. If the master database is down, users can still access the local copy of the database.

Is There a Business Need to Reduce Communication Costs?

Depending on the configuration (real time vs. nightly replication, etc.), there is a potential to reduce communications costs since the data access is local.

Is Scalability an Issue?

With users, data, and queries spread across multiple computers, scalability is less of a problem.

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 150 of 281

Can Users Benefit From the Increased Performance of Local Access?

Access to replicated data is fast since data is stored locally and users do not have to remotely access the master database. This is especially true for image and document data which cannot be quickly accessed from a central site. Making automatic copies of a database reduces locking conflicts and gives multiple sets of users better performance than if they shared the same database.

Product Considerations

What is the Current or Proposed Environment?

Platforms supported as well as source and target DBMS should be considered.

What are the Technical Requirements?

Products differ in features such as complete refresh vs. differential refresh (replication of changes), replication granularity (row, table, database), method of capturing changes (snapshot, SQL statement intercept, trigger-based, log-based), method of propagating copies (push, pull), propagation timing controls (database event-driven, scheduled based on interval, scheduled based on application event-driven, manually invoked), and conflict resolution mechanisms. Also important is what management utilities are available with the product.

Are Available Resources and Issue?

Products vary in the amount of resources required to install and operate the system.

What are the Business Requirements?

Three key considerations are:

- Who owns and uses the data? Replication products support one or more of the three ownership models: Primary site ownership—data is owned by one site; Dynamic site ownership—data owned by one site, however site location can change; and Shared site ownership—data ownership is shared by multiple sites.
- Which of the, four basic types of replication style is appropriate? The four styles are: Data dissemination—portions of centrally maintained data are replicated to the appropriate remote sites; Data consolidation—data is replicated from local sites to a central site where all local site data is consolidated; Replication of logical partitions—replication of partitioned data; and Update anywhere—multiple remote sites can possible update same data at same time.
- What is the acceptable latency period (amount of time the primary and target data can be out of synch)? There are three basic replication styles depending on the amount of latency that is acceptable: Synchronous—real-time access for all sites (no latency); Asynchronous near real-time—short period of latency for target sites; Asynchronous batch/periodic—predetermined period of latency for all sites.

Do I Already Have a Component That Satisfies This Criteria?

Many DBMS vendors ship replication products as either part of the base package or as an additional feature.

Possible Product Options

Sybase Replication Server; Oracle Symmetric Replication; CA-Ingres Replicator; InfoPump; DataPropagator Relational; Informix Replicator

Access **1408**

Access Services enable an application to retrieve data from a database as well as manipulate (insert, update, delete) data in a database. SQL is the primary approach for accessing records in today's database management systems.

Client-server systems often require data access from multiple databases offered by different vendors. This is often due to integration of new systems with existing legacy systems. The key architectural concern is in building the application where the multi-vendor problem is transparent to the client. This provides future portability, flexibility and also makes it easier for application developers to write to a single database access interface. Achieving database access transparency requires the following:

- Standards Based SQL API—this approaches uses a single, standards based set of APIs to access any database, and includes the following technologies: Open Database Connectivity (ODBC), Java Database Connectivity (JDBC), and Object Linking and Embedding (OLE DB).
- SQL Gateways provide a mechanism for clients to transparently access data in a variety of databases (e.g., Oracle, Sybase, DB2), by translating SQL calls written using the format and protocols of the gateway server or primary server to the format and protocols of the target database. Currently there are three contending architectures for providing gateway functions:
- Distributed Relational Data Access (DRDA) is a standard promoted by IBM for distributed data access between heterogeneous databases. In this case the conversion of the format and protocols occurs only once. It supports SQL**89** and a subset of SQL**92** standard and is built on top on APPC/APPN and TCP/IP transport stacks.
- IBI's EDA/SQL and the Sybase/MDI Open Server use SQL to access relational and non-relational database systems. They use API/SQL or T-SQL respectively as the standard interface language. A large number of communication protocols are supported including NetBIOS, SNA, DecNET, TCP/IP. The main engine translates the client requests into specific server calls. It handles security, authentication, statistics gathering and some system management tasks.

Implementation Considerations

Gateways may create bottlenecks, because all the clients go through a single gateway.

Security **1410**

Security Services enforce access control to ensure that records are only visible or editable by authorized people for approved purposes. Most database management systems provide access control at the database, table, or row level as well as concurrency control.

Implementation Considerations

Will the Application be Used in a Distributed Environment?

In a distributed environment, the need exists to provide access to the corporate data and resources in a secure and controlled manner. This access depends on the role of the user, the user group, etc. within that environment. Since security is an architecture component where functionality and robustness vary across engagements, the architectures usually provide a base set of security functions. These functions target securing the systems corporate data and resources, as opposed to securing an applications detailed functions.

The security component prevents unauthorized users from accessing corporate data/resources by providing the users with access codes—password & ID—that allows the user to login to the system or execute any (or a particular) application.

Security components can restrict access to functions within an application based on a users security level. The highest level security is whether the user has access to run the application. The next level checks if the user has access to functions within the application, such as service calls or windows. At an even lower level, the security component

could check security on more granular functions, such as widgets on a window.

Security usually resides on both the client and server platform in a distributed environment. True security should always be placed on the server platform, to protect the system through access outside of a client application.

Is There a Direct/Indirect Relationship Between the User Role/Group and the Data/Services?

There are situations where it is required for the system to maintain the relationship of the users role and the users access to specific system services/resources. For example, a database administrator will have read-write-delete access to the database, whereas a sales manager will have only read access to it for viewing the data in various forms. The security component should provide the functionality for validating the users resource access privileges based on the role of the user.

Indexing **1412**

Indexing Services provide a mechanism for speeding up data retrieval. In relational databases one or more fields can be used to construct the index. So when a user searches for a specific record, rather than scanning the whole table sequentially the index is used to find the location of that record faster.

Storage **1414**

Storage Services manage data physical storage. These services provide a mechanism for saving information so that data will live beyond program execution. Data is often stored in relational format (an RDBMS) but may also be stored in an object-oriented format (OODBMS) or other formats such as IMS, VSAM, etc.

Document Services **1416**

Document Services provide similar structure and control for documents that database management systems apply to record oriented data. A document is defined as a collection of objects potentially of different types (e.g., structured data, unstructured data, images, multi-media) a business user deals with. An individual document might be a table created using a spreadsheet package such as Microsoft Excel, a report created using a word processing package such as Lotus AmiPro, a Web page created using an HTML authoring tool, unstructured text or a combination of these object types Regardless of the software used to create and maintain the component parts, all parts together constitute the document, which is managed as a single entity.

Netcentric applications that are executed from a browser are particularly well suited for serving up document style information. If the Web application consists of more than just a few HTML documents, integration with a document management system should be considered. Document Services include: Storage Services, Indexing Services, Security Services, Access Services, Replication/Synchronization Services, and Versioning Services

Possible Product Options

Documentum Server; Saros; PC Docs

- Documentum—Documentum Enterprise Document Management System (EDMS) automates and accelerates the creation, modification, and reuse of business-critical documents, Web pages, and other unstructured data and all of the collaborative efforts involved.
- Saros—Saros Discovery Suite is the next generation client/server solution that integrates Saros Document Manager, FileNet Ensemble and Watermark Client to provide powerful, tightly-integrated electronic document management, workflow, and document-imaging capabilities.

Versioning **1418**

Versioning Services maintain a historical record of the changes to a document over time. By maintaining this record, these services allow for the re-creation of a document as it looked at any given point in time during it's evolution. Additional key versioning features record who made changes when and why they were made.

Replication/Synchronization **1404**

Replication Services support an environment in which multiple copies of documents must be maintained. A key objective is that documents should be shareable and searchable across the entire organization. Therefore, the architecture needs to provide logically a single repository, even though the documents are physically stored in different locations. The following might be possible if documents are replicated on alternative server(s): better availability or recoverability of a distributed application; better performance; reduced network cost; etc.

Synchronization Services perform the transactions required to make one or more information sources that are intended to mirror each other consistent. They support the needs of intermittently connected users or sites. Just like for databases, these services are especially valuable for users of mobile devices that need be able to work locally without a constant network connection and then be able to synchronize with the central server at a given point in time.

Implementation Considerations

Products such as Lotus Notes and Microsoft Exchange allow remote users to replicate documents between a client machine and a central server, so that the users can work disconnected from the network. When reattached to the network, users perform an update that automatically exchanges information on new, modified and deleted documents.

Note: Both Lotus Notes and MS Exchange provide a limited subset of the Document Services described in this section. This should be carefully evaluated when considering these products to provide document management services.

Access **1408**

Access Services support document creation, maintenance and retrieval. These services allow users to capture knowledge or content through the creation of unstructured information, i.e. documents. Access Services allow users to effectively retrieve documents that were created by them and documents that were created by others. Documents can be comprised of many different data types, including text, charts, graphics, or even audio and video.

Security **1410**

Documents should be accessed exclusively through the document management backbone. If a document is checked-in, check-out, routed, viewed, annotated, archived, or printed it should be done only by users with the correct security privileges. Those access privileges should be able to be controlled by user, role, and group. Analogous to record locking to prevent two users from editing the same data, document management access control services include check-in/check-out services to limit concurrent editing.

Indexing **1412**

Locating documents and content within documents is a more complex problem and involves several alternative methods. The Windows file manager is a simplistic implementation of a hierarchical organization of files and collection of files. If the user model of where documents should be stored and found can be represented in this way, the use of structure and naming standards can be sufficient. However, a hierarchical document filing organization is not suitable for many types of document queries (e.g., retrieving all sales order documents for over $1,000).

Therefore, most document management products provide index services that support the following methods for searching document repositories:

- Attribute Search—scans short lists (attributes) of important words that are associated with a document and returns documents that match the search criteria. For example, a user may query for documents written by a specific author or created on a particular date. Attribute search brings the capabilities of the SQL-oriented database approach to finding documents by storing in a database the values of specially identified fields within a document and a reference to the actual document itself. In order to support Attribute Search an index maintains documents' attributes, which it uses to manage, find and catalog documents. This is the least complicated approach of the searching methods.
- Full-text Search—searches repository contents for exact words or phrases and returns documents that match the search criteria. In order to facilitate Full-text Search, full-text indexes are constructed by scanning documents once and recording in an index file which words occur in which documents. Leading document management systems have full-text services built-in, which can be integrated directly into applications.
- Context Search—searches repository contents for exact words or phrases. Also, searches for related words or phrases by using synonyms and word taxonomies. For example, if the user searches for auto, the search engine should look for car, automobile, motor vehicle, etc.
- Boolean Search—searches repository contents for words or phases that are joined together using boolean operators (e.g., AND, OR, NOT). Same type of indexes are used for Boolean Search as for Full-Text Search.

The following products are used to index and search Web and non-Web documents:

- Verity Topic—delivers accurate indexing, searching and filtering of a wide variety of information sources and formats. Verity Topic is integrated directly into several document management products, allowing systems to full-text index its unstructured information. Verity Topic also offers a variety of products to help full-text index Web sites.
- Fulcrum—provides a variety of robust, multi-platform indexing and retrieval products that deliver full-function text retrieval capabilities. Fulcrums products are typically integrated with custom databases, Web sites and document management systems.

The Following Products are Mainly Used for Web Documents

- Microsoft Index Server 1.1—allows for search of Web documents, including Microsoft Word and Microsoft Excel. It works with Windows NT Server 4.0 and Internet Information Server 2.0 or higher to provide access to documents stored on an intranet or Internet site. Index Server supports full-text searches and retrieves all types of information from the Web browser including HTML, text, and all Microsoft Office documents, in their original format.
- Netscape Catalog Server 1.0—provides an automated search and discovery server for creating, managing, and keeping current an online catalog of documents residing on corporate intranets and the Internet. Catalog Server offers query by full text, category, or attributes such as title, author, date, etc. It also supports multiple file formats, including HTML, Word, Excel, PowerPoint, and PDF.

Storage **1414**

Storage Services manage the document physical storage. Most document management products store documents as objects that include two basic data types: attributes and content. Document attributes are key fields used to identify the document, such as author name, created date, etc. Document content refers to the actual unstructured information stored within the document. Generally, the documents are stored in a repository using one of the following methods:

- Proprietary database—documents (attributes and contents) are stored in a proprietary database (one that the vendor has specifically developed for use with their product).
- Industry standard database—documents (attributes and contents) are stored in an industry standard database such as Oracle or Sybase. Attributes are stored within traditional database data types (e.g., integer, character, etc.); contents are stored in the database's BLOB (Binary Large Objects) data type.
- Industry standard database and file system—Documents' attributes are stored in an industry standard database, and documents' contents are usually stored in the file-system of the host operating system. Most document management products use this document storage method, because today, this approach provides the most flexibility in terms of data distribution and also allows for greater scalability.

COMMUNICATION **1006,1008**

As illustrated in FIG. **15**, Network services provided by the Communications Services layer are grouped into four major categories of functionality: Virtual Resource, Directory, Messaging, and Security services **1502,1504, 1506,1508**.

Virtual Resource services proxy or mimic the capabilities of specialized, network connected resources. This allows a generic network node to emulate a specialized physical device. In this way, network users can interface with a variety of specialized resources.

Directory services play a key role in network architectures because of their ability to unify and manage distributed environments. Managing information about network resources involves a variety of processes ranging from simple name/address resolution to the logical integration of heterogeneous systems to create a common view of services, security, etc.

Messaging services transfer formatted information from one process to another. These services shield applications from the complexity of the network transport services.

Call centers and customer service centers are integral parts of many business operations. Call centers have enhanced business processes by managing telephone contact with potential customers, with the objective of improving the Quality of Service (QoS). Several customer and business drivers are motivating a transition from traditional cost-based call centers to more strategic centers focused on customer interaction.

Communications Security services control access to network-attached resources. Combining network Security services with security services in other parts of the system architecture (e.g., application and database layers) results in robust security.

Implementation Considerations

Is Data Translation Required?

Communications middleware can translate data into a format that is compatible with the receiving process. This may be required in a heterogeneous environment. An example is data translation from ASCII-to-EBCDIC. It is important to note that data translation may not be provided by all middleware products.

Are Additional Communications Services Required?

Communications middleware can provide additional communications services that may be required by the applications. Additional services include dynamic message routing, guaranteed delivery, broadcasting, queuing, and priority delivery. These common services are usually provided in the communications middleware rather than addressing them in each application separately. Different communications middleware products provide different services. Additionally, many middleware packages, such as Tuxedo, provide OLTP functionality.

Is a Packaged Middleware Solution Desired?

Depending on the functionality required, communications middleware can be very complex to custom develop. In addition, products have evolved to a point where proven solutions exist. Based on this, it can be desirable to buy communications middleware rather than to build it. Considerations of time, budget, skills, and maintenance should be taken into account when selecting between a packaged middleware product and custom developed middleware. In some instances, custom developed middleware may still be preferred.

What is the Clients Middleware Direction?

There is a definite functionality overlap between communications middleware and several other middleware components such as transaction services and information access. In addition, communications middleware may be provided by various CASE tools. An example of this is the Distribution Services component of FCP. Because of this overlap, it is important to understand the clients overall direction toward middleware and the specific middleware functionality required by the overall solution.

Is a Simplified Developers Interface Important?

The simplified interface associated with communications middleware can help to reduce the complexity of developing Netcentric applications. The simplified interface helps reduce the development complexity by insulating the business applications from the network protocols. Because of this, application developers do not need to understand the intricacies and somewhat cryptic APIs associated with network transport protocols.

Is Location Transparency Required?

Communication middleware allows the client application to access any service on any physical server in the network without needing to know where it is physically located. This capability may be required in an environment with many physical servers or in an environment that is very dynamic. It is important to note that location transparency may not be provided by all middleware products.

Does the Application Need to run on Multiple Platforms?

Communications middleware is designed to allow applications to access various transport protocols from various vendors. From a network interface perspective, it should be easier to port an application from one computing platform to another if the application is using communications middleware. Of course, other porting issues will need to be considered.

Virtual Resources **1502**

Virtual Resource services proxy or mimic the capabilities of specialized, network-connected resources. This allows a generic network node to emulate a specialized physical device. In this way, network users can interface with a variety of specialized resources. An examples of a Virtual Resource service is the capability to print to a network printer as if it were directly attached to a workstation.

Fax **1510**

Fax Services provide for the management of both in-bound and out-bound fax transmissions. If fax is used as a medium for communicating with customers or remote employees, in-bound fax services may be required for centrally receiving and electronically routing faxes to the intended recipient. Out-bound fax services can be as simple as supporting the sharing on the network of a single fax machine or group of machines for sending faxes.

Fax services can provide centrally managed faxing capabilities, thus eliminating the need for fax modems on every workstation. A fax server generally provides Fax services to clients, such as receiving, queuing, and distributing incoming faxes and queuing and sending outgoing faxes. Clients can view faxes and generate faxes to be sent.

Applications may compose and transfer faxes as part of notifying users or delivering information. For example, an application may use Fax services to add customer-specific information to a delivery receipt form and fax the form to a customer.

Implementation Considerations

More sophisticated out-bound fax architecture services are required for supporting fax-back applications. Fax-back applications, when coupled with Computer Telephone Integration (CTI) are popular for automating customer requests for product or service information to be faxed to them.

Possible Product Options

Cheyenne Softwares Faxserve; Lotus Fax Server for Lotus Notes; Sirens Siren Fax

The Following are Examples of Fax Servers:

The Lotus® Fax Server (LFS)—provides fax services to users working on a network running NotesMail®. In addition to combining outgoing and incoming fax capabilities in a single product, the LFS provides additional features, such as automatic routing, and print-to-fax driver software that extends fax capabilities to any Windows-based Notes client. The LFS supports a wide variety of fax modems, fax cards and fax file formats through the incorporation of device technologies from Optus Software, Inc.

Cheyenne Software's Faxserve

The Following is an Example of a Product That Allows Applications to Generate Faxes:

Siren's Siren Fax

File Sharing **1512**

FIG. **16** illustrates File Sharing services **1512**. File Sharing services allow users to view, manage, read, and write files that may be located on a variety of platforms in a variety of locations. File Sharing services enable a unified view of independent file systems. This is represented in FIG. **16**, which shows how a client can perceive remote files as being local.

File Sharing services can provide the following capabilities:

Transparent access—access to remote files as if they were local

Multi-user access—distribution and synchronization of files among multiple users, including file locking to manage access requests by multiple users

File access control—use of Security services (user authentication and authorization) to manage file system security

Multi-platform access—access to files located on various platforms (e.g., UNIX, NT, etc.)

Integrated file directory—a logical directory structure that combines all accessible file directories, regardless of the physical directory structure

Fault tolerance—use of primary and replica file servers to ensure high availability of file system

Scalability—ability to integrate networks and distributed file systems of various sizes

Possible Product Options

Novell's NetWare/IntranetWare; Microsoft's Windows NT Server; Sun Microsystems NFS and WebNFS; Novell's IntranetWare NFS Services; IBM/Transarcs Distribute File System (DFS); Transarc's AFS

The Following are Examples of File Sharing Products:

Novell's NetWare/IntranetWare—Novell's NetWare network operating system includes distributed file services, supported by the NetWare Core Protocol (NCP). NetWare Directory Services (NDS) manages naming and security for files on distributed platforms.

Microsoft's Windows NT Server

Server Message Block (SMB)—native file-sharing protocol in Windows 95, Windows NT, and OS/2.

Common Internet File System (CIFS)—an enhancement to SMB for distributed file systems in a TCP/IP environment.

Distributed File System (Dfs)—a utility for Windows NT Server that provides file services in a Microsoft environment.

Network File System (NFS)—NFS is a native UNIX file access protocol and is also available as an operating system add-on product that provides distributed file services. Sun Microsystems introduced NFS in 1985. NFS has been widely adopted and has been ported to a variety of platforms.

The Following are Examples of Products That Provide NFS Services.

Sun Microsystems' NFS and WebNFS Novell's IntranetWare NFS Services

AFS—A distributed file system for distributed UNIX networks; derived from Carnegie-Mellon University's Andrew File System. Similar to NFS, but differs in terms of the name space, system performance, security, etc. AFS is distributed by Transarc.

IBM/Transarc's Distribute File System (DFS)—a scaleable distributed file system that offers replication, security, etc.

Paging **714**

Wireless short messaging (i.e., paging) can be implemented through wireless systems such as paging networks, GSM voice/data networks, PCS voice/data networks, and dedicated wireless data networks. Paging virtual resource services provide the message formatting and display functionality that allows network nodes to interface with wireless paging systems. This service emulates the capabilities of one-way and two-way pagers. Paging systems allow pages to be generated in various ways:

E-mail messages to a specified mailbox

DTMF (touch tone) signaling to a voice response system

Encoded digital messages transferred into a paging provider gateway

Messages transferred to a locally attached two-way wireless pager

Possible Product Options

TelAlert; e-mail Systems

e-mail systems—some e-mail systems and fax servers can be configured to generate pages to notify users when a defined event occurs such as e-mail/fax arriving.

Telamon's TelAlert—TelAlert provides notification capabilities for UNIX systems. For example, it can page support personnel in the event of system problems.

Phone **1516**

Phone virtual resource services extend telephony capabilities to computer platforms. For example, an application on a desktop computer can place and receive telephone calls for the user. Phone virtual resource services may be used in customer care centers, help desks, or any other environment in which it is useful for a computer to replace a telephone handset.

Phone services enable clients, servers, and specialized telephony nodes (PBXs, ACDs, etc.) to control the telephony environment through the following telephony controls:

Call control:

Controls telephone features

Controls recorded messages

Manipulates real time call activities (e.g., make call, answer, transfer, hold, conference, mute transfer, release, route call, call treatments and digits collected)

Telephone status control

Controls telephone status functions

Logs users in and out of the system

Sets ready, not ready, and make busy statuses for users

The following are examples of uses of Phone virtual resources:

PC Telephony—PC telephony products allow desktop computers to act as conduits for voice telephone calls.

Internet Telephony—Internet telephony products enable voice telephone calls (and faxing, voice mail retrieval, etc.) through the Internet. For example, an Internet telephony product can accept voice input into a workstation, translate it into an IP data stream, and route it through the Internet to a destination workstation, where the data is translated back into audio.

Desktop Voice Mail—Various products enable users to manage voice mail messages using a desktop computer.

Possible Product Options

Lucent PassageWay; COM2001s TransCOM; NetSpeaks WebPhone; VocalTecs Internet Phone; IDTs Net2Phone; Octel Communications Unified Messenger

The following are examples of vendors that provide PC telephony products:

Lucent PassageWay—suite of products that connect PCs to PBXs.

COM2001's TransCOM—voice, data and call-management system (dialing, voice mail, faxing, voice recognition, caller ID, etc.) for personal computers.

The following are examples of Internet telephony products:

NetSpeak's WebPhone

VocalTec's Internet Phone

IDT's Net2Phone

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 155 of 281

The following is an example of a desktop voice mail product:

Octel Communication's Unified Messenger

Terminal **1518**

Terminal services allow a client to connect to a non-local host via a network and to emulate the profile (e.g., the keyboard and screen characteristics) required by the host application. For example, when a workstation application logs on to a mainframe, the workstation functions as a dumb terminal. Terminal Services receive user input and send data streams back to the host processor. If connecting from a PC to another PC, the workstation might act as a remote control terminal (e.g., PCAnywhere).

The following are examples of Terminal services:

Telnet—a simple and widely used terminal emulation protocol that is part of the TCP/IP communications protocol. Telnet operates establishing a TCP connection with the remotely located login server, minicomputer or mainframe. The client's keyboard strokes are sent to the remote machine while the remote machine sends back the characters displayed on the local terminal screen.

3270 emulation—emulation of the 3270 protocol that is used by IBM mainframe terminals.

tn3270—a Telnet program that includes the 3270 protocol for logging onto IBM mainframes; part of the TCP/IP protocol suite.

X Window System—allows users to simultaneously access applications on one or more UNIX servers and display results in multiple windows on a local display. Recent enhancements to XWS include integration with the Web and optimization of network traffic (caching, compression, etc.).

Remote control—While terminal emulation is typically used in host-based environments, remote control is a sophisticated type of client/server Terminal service. Remote control allows a client computer to control the processing on a remote desktop computer. The GUI on the client computer looks as if it is the GUI on the remote desktop. This makes it appear as if the remote applications are running on the client.

rlogin—a remote terminal service implemented under BSD UNIX. The concept behind rlogin is that it supports "trusted" hosts. This is accomplished by having a set of machines that share common file access rights and logins. The user controls access by authorizing remote login based on a remote host and remote user name.

Possible Product Options

Hummingbird's Exceed; Network Computing Devices' PC-Xware; Citrix WinFrame; Carbon Copy; pcANYWHERE; Stac's Reachout; Traveling Software's LapLink

The following are examples of X Window System products:

Hummingbird's Exceed

Network Computing Devices' PC-Xware

The following are examples of remote control products:

Citrix's WinFrame

Microcom's Carbon Copy

Symantec's pcANYWHERE

Stac's Reachout

Traveling Software's LapLink

Printing **1520**

Print services connect network workstations to shared printers. The administration of Print Services is usually handled by a print server. Depending on the size of the network and the amount of resources the server must manage, the print server may run on a dedicated machine or on a machine that performs other server functions. A primary function of print servers is to queue print jobs sent to network printers. The queued jobs are stored in a print buffer on the print server and are sent to the appropriate network printer as it becomes available. Print services can also provide the client with information including print job status and can manage in-progress print jobs.

Possible Product Options

Novell's Netware Distributed Print Services (NDPS); Novell's Netware UNIX Print Services; Microsoft; Windows NT Server; Line Printer Daemon (LPD)

The following are examples of print server products:

Novell's Netware Distributed Print Services (NDPS)—provides central management of print services for NetWare networks.

Novell's Netware UNIX Print Services—a supplement to Novell's NetWare 4.1 server which allows NetWare and UNIX clients to share UNIX or Netware printers.

Microsoft Windows NT Server—provides central management of print services for NT networks.

Line Printer Daemon (LPD)—UNIX print management facilities, which include client and server utilities for spooling print jobs. Related programs include lpr (sends print job to spool) and lp (sends request to printer).

Audio/Video **1522**

Audio/Video services allow nodes to interact with multimedia data streams. These services may be implemented as audio-only, video-only, or combined audio/video:

Audio services—Audio services allow components to interface with audio streams such as the delivery of music or radio content over data networks.

Video services—Video services allow components to interface with video streams such as video surveillance. Video services can add simple video monitor capabilities to a computer, or they can transform the computer into a sophisticated video platform with the ability to generate and manipulate video.

Combined Audio/Video services—Video and audio content is often delivered simultaneously. This may be accomplished by transferring separate audio and video streams or by transferring a single interleaved stream. Examples include video conferencing and television (traditional or interactive).

Audio/Video services can include the following functionality:

Streams content (audio, video, or both) to end users

Manages buffering of data stream to ensure uninterrupted viewing/listening

Performs compression and decompression of data

Manages communications protocols to ensure smooth delivery of content Manages library of stored content and/or manages generation of live content

Audio/Video services draw upon lower-level services such as streaming and IP Multicast in order to efficiently deliver content across the network.

Possible Product Options

Progressive Networks RealVideo; Microsoft's NetShow; Vxtremes Web Theater; Intels ProShare; Creative Labs Video WebPhone

The following products are examples of video servers:

Progressive Networks' RealVideo

Microsoft's NetShow

Vxtreme's Web Theater

The following products are examples of video conferencing systems:

Intel's ProShare

Creative Labs' Video WebPhone

Directory Services **1504**

A full-featured Directory Service organizes, categorizes and names networked resources in order to provide a comprehensive picture of clients, servers, users, applications and other resources. The service typically includes a database of objects, representing all nodes and resources on a network. The database manages relationships between users and networks, network devices, network applications, and information on the network. The Directory service can organize network nodes to reflect the topology and organization of the enterprise and its policies. The Directory service makes resources location and platform independent, since it allows users to locate resources via the directory and regardless of their physical location. The Directory service also maps between logical resource names (e.g., "Marketing_Printer") and physical resource address (e.g., 10.27.15.56). (See Name service, below).

Directory service products utilize Security services to track access rights for access to network resources and information. The Directory service is an efficient way to manage resource security, since the directory offers a logical representation of all resources in the enterprise. In addition, the Directory service can act as a single point of entry into the network, meaning users can receive access to allowed resources by authenticating themselves a single time to the Directory service. (For more information on authentication and authorization, refer to the Comm. Security service.)

In summary, the Directory service performs the following functions:

Stores information about network resources and users and tracks relationships Organizes resource access information in order to aid resources in locating and accessing other resources throughout the network

Provides location transparency, since resources are accessed through a directory rather than based on their physical location

Converts between logical resource names and physical resource addresses Interacts with Security services such as authentication and authorization track identities and permissions

Provides single network logon to file and print resources; can provide single network logon for network applications that are integrated with the Directory service

Distributes directory information throughout the enterprise (for reliability and location-independent access)

Synchronizes multiple directory databases

Enables access to heterogeneous systems (integration of various network operating systems, platforms, etc.)

Directory Standards—There are a variety of standards for directories. Vendor-specific directory products build upon (and extend) standards to provide a robust, full-featured enterprise directory.

The following are examples of standards related to Directory services:

X.500 an ITU-T standard for a hierarchical directory containing user and resource information; includes Directory Access Protocol (DAP), which can be used to access directory information.

Lightweight Directory Access Protocol (LDAP) a de facto standard for accessing X.500-compatible directory information in an Internet/intranet environment.

Implementation Considerations

One of the most popular network directory services is Novell Directory Services (NDS) used with Netware 4.x. This system allows users to access services and resources with a single login, regardless of where the user location is or where the resource location is. Another example of a directory service is the ISO X.500 standard. This method is not widely used due to its high overheads. In addition to these two protocols, Windows NT uses a similar system called Primary Domain Control. This system allows for the same type of directory mapping as NDS and X.500.

Another protocol that has emerged is the Lightweight Directory Access Protocol (LDAP), which is a slimmed-down version of the X.500 directory client and is seen as a possible replacement for X.500. LDAP is a standard protocol for accessing and updating directory information in a client/server environment; it has evolved into an emerging standard for directory replication for the Internet, and is backed by vendors such as Netscape, Novell, Microsoft, IBM and AT&T that can provide low-level compatibility among directory systems.

Another helpful feature to look out for is support for dynamic IP addressing via DHCP. This lets the router handle the process of sharing a small number of IP addresses among the members of the workgroup. Support for dynamic IP addressing is now part of Windows 95 and Macintosh System 7.6, among other operating systems.

Possible Product Options

Novells Netware Directory Service; Netscapes Directory Server; Microsofts Active Directory; Banyan Systems StreetTalk

The following are examples of products that provide full-featured Directory services.

Novell's Netware Directory Service

Netscape's Directory Server

Microsoft's Active Directory Banyan Systems' StreetTalk

The following is an example of a meta-directory product:

Zoomit VIA—integrates network operating system directories, application databases, and human resource databases (includes Lotus cc:Mail, Lotus Notes, Novell NDS, Microsoft NT Domain Controller and Active Directory, Microsoft Exchange, Banyan VINES, Netscape Directory Server), thus allowing unified access and maintenance.

The following are examples of Name services:

Domain Name Service—The most common and widely used Name Service on the Internet is Domain Name Service (DNS) which resolves a pronounceable name into an IP address and vice versa. For instance, DNS could resolve the domain name of www.ac.com to be 204.167.146.195. DNS functionality is distributed across many computers within the network.

Microsoft's Windows Internet Name Service (WINS)—WINS is Microsoft's proprietary method for mapping IP addresses to NetBIOS device names. WINS works with Windows 3.x, Windows 95, and Windows NT clients.

The following are examples of products that provide Domain services:

Network Information Service (NIS)—Developed and licensed by Sun Microsystems for use in UNIX environments, NIS tracks user names, passwords, user IDs, group IDs, and host names (along with other system files) through a centralized NIS database.

Microsoft's Windows NT Server Domain Controller

Domain Services **1524**

A network domain is a set of network nodes under common control (i.e., common security and logins, unified addressing, coordinated management, etc.). Domain services manage these types of activities for the network nodes in a domain. Domain services may be limited in their ability to support heterogeneous systems and in the ability to scale to support the enterprise.

Name Service **1526**

The Name service creates a logical "pronounceable" name in place of a binary machine number. These services could be used by other communications services such as File Transfer, Message Services, and Terminal Services. A Name service can be implemented on its own, or as part of a full-featured Directory service.

Core Messaging **1528**

Broadly defined, Messaging services enable information or commands to be sent between two or more recipients. Recipients may be computers, people, or processes within a computer. Messaging Services are based on specific protocols. A protocol is a set of rules describing, in technical terms, how something should be done. Protocols facilitate transport of the message stream. For example, there is a protocol describing exactly what format should be used for sending specific types of mail messages. Most protocols typically sit "on top" of the following lower level protocol:

TCP/IP—Transmission Control Protocol/Internet Protocol (TCP/IP) is the principle method for transmitting data over the Internet today. This protocol is responsible for ensuring that a series of data packets sent over a network arrive at the destination and are properly sequenced.

Messaging services transfer formatted information from one process to another. By drawing upon Messaging services, applications can shield themselves from the complexity of the low-level Transport services. The Core Messaging services category includes styles of messaging that support basic inter-process communication (IPC). There are a variety of architecture options used to support IPC. They can be divided into Store and Forward, Synchronous and Asynchronous Message Services.

Store and Forward Message Services—provide deferred message service processing. A Store and Forward Message Service may use an E-Mail infrastructure upon which to build applications. Common uses would be for forms routing and E-mail.

Synchronous Message Services—allow an application to send a message to another application and wait for a reply before continuing. Synchronous messaging is typically used for update and general business transactions. It requires time-out processing to allow the application to re-acquire control in the event of failure.

Asynchronous Message Services allow an application to send a message to another application and continue processing before a reply is received. Asynchronous messaging is typically used for larger retrieval type processing, such as retrieval of larger lists of data than can be contained in one message.

Additionally, inter-process messaging services are typically one of two messaging types:

Function Based—uses the subroutine model of programming. The message interface is built upon the calling program passing the appropriate parameters and receiving the returned information.

Message Based—message-based approach uses a defined message format to exchange information between processes. While a portion of the message may be unstructured, a defined header component is normally included. A message-based approach is not limited to the call/return structure of the function-based model and can be used in a conversational manner.

Core Messaging services are categorized by the characteristics of the information being transferred:

File Transfer

RPCs

Message-Oriented Middleware

Streaming

How do Messaging services compare to Transaction Processing (TP) services? TP services offer broad functionality to support application management, administrative controls, and application-to-application message passing. TP services may include global transaction coordination, distributed two-phase commit, database support, coordinated recovery after failures, high availability, security, and work load balancing. TP services may utilize Messaging services, which provide basic interprocess communication.

Another category of Messaging services, Specialized Messaging services, includes services that extend Core Messaging services to provide additional functionality.

Implementation Considerations

Is Guaranteed Delivery Required?

RPCs do not support guaranteed message delivery techniques such as store-and-forward and queuing. Consequently, RPCs depend upon the availability of the physical network and server processes. Therefore, network stability is important to consider when deciding to use RPCs.

How Important is Flexibility?

In general, RPCs work best with tightly coupled applications or in environments where significant application modifications are unlikely. RPCs may be desirable if the application being developed is intended to be shrink wrapped and sold.

Is Synchronous or Asynchronous Program Control Required?

Function based middleware such as RPCs traditionally provide synchronous program control. Therefore, they tend to pass control from the client process to the server process. When this occurs, the client is dependent on the server and must wait to perform any additional processing until the servers response is received. This type of program control is also known as blocking. Some RPC vendors are enhancing their products to support asynchronous program control as well.

What Type of Conversation Control is Required?

RPCs permit one side of the conversation (the client) to only make requests, while the other side (the server) may only make replies. Conversation control is passed from the client to the server since the client, for each request, causes one or more functions to execute on the server while it waits for its reply. With RPCs, developers do not need to be concemed with the state of the conversation between the client and the server. In most cases, the absence of conversation states simplifies the design and development effort.

Is Yclient Interested in a Stable or Emerging Technology?

RPCs have existed for many years and are considered to be a mature, stable, proven solution.

Is it Important to Minimize Development Complexity?

Due to the synchronous program control and the request/reply conversation control, RPCs can be fairly straightforward to design and build. The complexity is also reduced since RPC calls are completely independent of any previous or future RPC call. On the other hand, RPCs usually require a specific RPC compiler, which may add to the development complexity.

Are Extended Technical Capabilities Required?

If any of the following capabilities are required, message based middleware should be considered. It may also be possible to incorporate these capabilities into a function

based middleware solution, but significant custom modification and development may be required.

Guaranteed Delivery

Store and Forward

Queuing

Priority Message Delivery

Dynamic Routing

Multicasting and Broadcasting

Load Balancing

Product Considerations

What are the Client's Budgetary Constraints?

Costs may vary greatly among middleware products. There are many factors to consider when looking at middleware. To begin, middleware products can require extensive consulting and support services just to install. Therefore, understanding the set-up and configuration costs are important. There are also additional products required to complete an environment such as additional networking software which may be necessary for each individual client. In addition, development seat costs and production seat costs must considered.

Is Synchronous or Asynchronous Communications Required?

All RPC products support synchronous program control. Some vendors are enhancing their products to provide asynchronous capabilities as well. Asynchronous means that while information is being passed via send and receive commands, programs can continue to process other tasks while waiting for a response to a request.

What's the Clients Position on DCE?

DCE software, developed by Open Systems Foundation (OSF), is licensed to OSF-member companies to form products that provide common services. The RPC is one of several DCE common services. Some clients may desire to be aligned with DCE-based solutions.

Is the Middleware Compatible with the Other Technology Architecture Components?

Communications middleware products must integrate with other technology architecture components, development tools, and operations tools. Therefore, it is necessary to understand the compatibility between these tools and the communications middleware product.

Is it Important for the Product to Support Multiple Platforms and Operating Systems?

The middleware products must support the required computing platformn such as Windows, UNIX, and Mainframe. It is common for vendors to claim that their product supports various platforms and operating systems, when in reality, that platform and operating system may be supported in a future release. It is important to request references of implementations of the platforms and operating systems that are important to your specific environment.

What is the Client's Vendor Direction?

When evaluating a middleware product, its important to consider the clients relationships with vendors in the technology market. For example, if the client has a strong relationship with a vendor who is also in the middleware market, it would be wise to investigate and consider such a vendor for the clients middleware solution.

Is it Important for the Product to Support Multiple Network Protocols?

The middleware products must support the network protocols such as TCP/IP, LU6.2, and IPX/SPX that are important to your specific environment. It is important to note that protocols can vary across platforms. Ensure that the clients specific transport protocol version is supported by the communications middleware product. For example, communications middleware vendors may support TCP/IP but they may not support the particular TCP/IP vendor that the client has selected.

Is a Quick Response Time Critical?

RPC performance may vary between products based upon the internal mechanisms and techniques of the product. For example, slow performance may be due to the processing overhead associated with each RPC call. Some RPC products may improve performance by utilizing special techniques used to invoke the server every time a client request arrives. Performance should be considered as a product differentiator.

What Level of Security is Required?

There are potential security issues associated with the execution of commands on a remote system. Some vendors install security features into their products. It is also possible for the architecture team to build additional security into the overall solution.

Is Yclient Interested in a Stable or Emerging Product?

Vendors should be evaluated on the quality of service they offer, their market share, the age of their product, the installed base of their product, and their financial stability. In addition, since this market is still emerging, there are many small vendors in the market trying to offer solutions. Vendor and product stability should be taken very seriously.

File Transfer **1530**

File Transfer services enable the sending and receiving of files or other large blocks of data between two resources. In addition to basic file transport, features for security, guaranteed delivery, sending and tracking sets of files, and error logging may be needed if a more robust file transfer architecture is required. The following are examples of File Transfer standards:

File Transfer Protocol (FTP) allows users to upload and download files across the network. FTP also provides a mechanism to obtain filename, directory name, attributes and file size information. Remote file access protocols, such as Network File System (NFS) also use a block transfer method, but are optimized for online read/write paging of a file. HyperText Transfer Protocol (HTTP)—Within a Web-based environment,

Web servers transfer HTML pages to clients using HTTP. HTTP can be thought of as a lightweight file transfer protocol optimized for transferring small files. HTTP reduces the inefficiencies of the FTP protocol. HTTP runs on top of TCP/IP and was developed specifically for the transmission of hypertext between client and server. The HTTP standard is changing rapidly.

Secure Hypertext Transfer Protocol (S-HTTP)—a secure form of HTTP, mostly for fi nancial transactions on the Web. S-HTTP has gained a small level of acceptance among merchants selling products on the Internet as a way to conduct financial transactions (using credit card numbers, passing sensitive information) without the risk of unauthorized people intercepting this information. S-HTTP incorporates various cryptographic message formats such as DSA and RSA standards into both the Web client and the Web server.

File Transfer and Access Management (FTAM)—The OSI (Open Systems Interconnection) standard for file transfer, file access, and file management across platforms.

Implementation Considerations

Additional options for File Transfer Services in a homogeneous environment could include the native operating systems copy utility, i.e. Windows NT Copy features.

Possible Product Options

Computer Associates CA-XCOM; RemoteWare; Hewlett-Packards HP FTAM; IBMs Files On-Demand gateway

The following are examples of File Transfer products:

Computer Associates CA-XCOM; RemoteWare; Hewlett-Packards HP FTAM; IBMs Files On-Demand gateway

The following are examples of File Transfer products:

Computer Associates' CA-XCOM—provides data transport between mainframes, midrange, UNIX, and PC systems. XcelleNet's Remote-Ware—retrieves, appends, copies, sends, deletes, and renames files between remote users and enterprise systems. Hewlett-Packard's HP FTAM—provides file transfer, access, and management of files in OSI networks.

The following product provides File Transfer translation:

IBM's Files On-Demand gateway—acts as a gateway between Web-based and mainframe-based FTP services to allow users to download mainframe-based files from a World Wide Web browser.

RPC **1532**

RPCs (Remote Procedure Calls) are a type of protocol by which an application sends a request to a remote system to execute a designated procedure using the supplied arguments and return the result. RPCs emulate the function call mechanisms found in procedural languages (e.g., the C language). This means that control is passed from the main logic of a program to the called function, with control returning to the main program once the called function completes its task. Because RPCs perform this mechanism across the network, they pass some element of control from one process to another, for example, from the client to the server. Since the client is dependent on the response from the server, it is normally blocked from performing any additional processing until a response is received. This type of synchronous data exchange is also referred to as blocking communications.

Possible Product Options

Sun Microsystems ONC+; OpenGroups DCE RPC; Novells NetWare RPC; NobleNet's EZ-RPC; Transarcs DCE RPC; Microsofts Windows95/NT RPC

Sun Microsystems' ONC (Open Network Computing)

OpenGroup's DCE (Distributed Computing Environment)

Novell's NetWare RPC NobleNet EZ-RPC Transarc's DCE

Microsoft's Windows95/NT RPC

Message Oriented **1534**

Message-Oriented Middleware (MOM) refers to the process of distributing data and control throughout the exchange of records known as messages. MOM provides the application developer with a set of simple verbs (e.g., connect, send, receive, and disconnect) that are used to exchange information with other distributed applications.

Message-Oriented Middleware is responsible for managing the interface to the underlying communications architecture via the communications protocol APIs and ensuring the delivery of the information to the remote process. This interface provide the following capabilities:

Translating mnemonic or logical process names to operating system compatible format

Opening a communications session and negotiating parameters for the session

Translating data to the proper format

Transferring data and control messages during the session

Recovering any information if errors occur during transmission

Passing results information and status to the application.

An application continues processing after executing a MOM request, allowing the reply to arrive at a subsequent time. Thus, unlike RPCs, MOM implements a "non-blocking" or asynchronous messaging architecture.

Message-Oriented Middleware products typically support communication among various computing platforms (e.g., DOS, Windows, OS/2, Macintosh, UNIX, and mainframes).

There are three types of Message-Oriented Middleware commonly implemented:

Message Passing

Message Queuing

Publish and Subscribe

Message Passing—as illustrated in FIG. **17**, is a direct, application-to-application communication model. An application request is sent in the form of message from one application to another. The communication method can be either synchronous like RPCs or asynchronous (through callback routines). In a message-passing model, a direct link between two applications that participate in the message exchange is always maintained.

Message Queuing (also known as Store and Forward)—as depicted in FIG. **18**, is an indirect application to application communication model that allows applications to communicate via message queues, rather than by calling each other directly. Message queuing is asynchronous by nature and connectionless, meaning that the recipient need not be directly available when the message is sent. Moreover, it implies support for reliable, guaranteed and assured (non-duplicate) message delivery.

Publish and Subscribe (also known as Push messaging)—as shown in FIG. **19**, is a special type of data delivery mechanism that allows processes to register an interest in (i.e., subscribe to) certain messages or events. An application then sends (publishes) a message, which is then forwarded to all processes that subscribe to it.

Implementation Considerations

When trying to decide whether to use MOM technology, keep the following characteristics of this type of middleware in mind:

MOMs are high speed, generally connectionless and are usually deployed for executing applications with a nonblocking sender

MOM solutions are especially useful for inter-application communication and are increasingly popular for inter-enterprise work

MOMs support end-to-end business applications and process inter-operability

MOMs are designed for heavily used production applications and are generally capable of high throughput rates and fast transfer times. Data is usually forwarded immediately, although it is possible to store it for later processing

Possible Product Options

PeerLogics PIPES; IBM MQSeries; BEAs MessageQ; Momentum XIPC; Microsoft MQ (Falcon); TibCo's Rendezvous

Message Passing

PeerLogic's PIPES

PIPES Platform applications communicate through a messaging interface that allows asynchronous, non-blocking communications. The messaging model is well suited to complex multi-tier applications because it inherently supports asynchronous, event-driven communications.

Message Queuing

IBM's MQSeries

New features found in version 5 include:

A new Internet gateway that allows customers and partners to run mission critical business applications over an unreliable network.

Enhanced message distribution carries more business information, while minimizing use of networks.

Performance improvements gives message transmission at least 8 times faster than previous versions

Resource Coordination ensures that data held in databases is always updated completely—or not at all, if processing cannot complete.

Additional developer features include further language support for C++, Java and PL/1, and interoperability with current and previous MQSeries versions.

Easier implementation because MQSeries now has the same install and use characteristics as other IBM Software Servers.

BEA's MessageQ

Key highlights of the MessageQ product include:

High performance-up to thousands of non-recoverable messages/second; hundreds of recoverable messages/second

Both synchronous, and asynchronous message delivery

Broadest platform support in the industry including UNIX, Windows NT, OpenVMS, and mainframes

Common Application Programming Interface (API)

Publish and subscribe (broadcasting)

Microsoft Windows client product with support for DLLs (Dynamically Linked libraries), Visual Basic, and Power Builder development environments

Message recovery on all BEA MessageQ clients and servers

Interoperability with IBM MVS/CICS and IBM MVS/IMS

Large message size—up to 4 MB—eliminates need for message partitioning

Momentum's XIPC

XIPC is an advanced software toolset for the development of multitasking and distributed applications. XIPC provides fault-tolerant management of guaranteed delivery and real-time message queuing, synchronization semaphores and shared memory, all of which are network-transparent.

Microsoft Message Queue Server (MSMQ, formerly known as Falcon)

Publish and Subscribe

TibCo's Rendezvous

TIB/Rendezvous' publish/subscribe technology is the foundation of TIBnet, TibCos solution for providing information delivery over intranets, extranets and the Internet. It is built upon The Information Bus® (TIB®) software, a highly scaleable messaging middleware technology based on an event-driven publish/subscribe model for information distribution. Developed and patented by TIBCO, the event-driven, publish/subscribe strategy allows content to be distributed on an event basis as it becomes available. Subscribers receive content according to topics of interest that are specified once by the subscriber, instead of repeated requests for updates. Using IP Multicast, TIBnet does not clog networks, but instead, provides for the most efficient real-time information delivery possible.

Streaming **1536**

Streaming is the process of transferring time-sensitive data streams (e.g., video and/or audio) in real-time. Streaming differs from the other types of Core Messaging services in that it delivers a continuous, one-way stream of data, rather than the relatively short messages associated with RPC and Message-Oriented Middleware messaging or the large, batch transfers associated with File Transfer. (While the media stream is one-way from the server to the client, the client can issue stream controls to the server.) Streaming may be used to deliver video, audio, and/or other real-time content across the Internet or within enterprise networks.

Streaming is an emerging technology. While some multimedia products use proprietary streaming mechanisms, other products incorporate standards. The following are examples of emerging standards for streaming protocols. Data streams are delivered using several protocols that are layered to assemble the necessary functionality.

Real-time Streaming Protocol (RTSP)—RTSP is a draft Internet protocol for establishing and controlling on-demand delivery of real-time data. For example, clients can use RTSP to request specific media from a media server, to issue commands such as play, record and pause, and to control media delivery speed. Since RTSP simply controls media delivery, it is layered on top of other protocols, such as the following.

Real-Time Transport Protocol (RTP)—Actual delivery of streaming data occurs through real-time protocols such as RTP. RTP provides end-to-end data delivery for applications transmitting real-time data over multicast or unicast network services. RTP conveys encoding, timing, and sequencing information to allow receivers to properly reconstruct the media stream. RTP is independent of the underlying transport service, but it is typically used with UDP. It may also be used with Multicast UDP, TCP/IP, or IP Multicast.

Real-Time Control Protocol (RTCP)—RTP is augmented by the Real-Time Control Protocol. RTCP allows nodes to identify stream participants and communicate about the quality of data delivery.

The following table summarizes the protocol layering that supports Streaming:

functionality	sample protocol options	architecture service
controlling media delivery	RTSP or proprietary	Streaming Messaging service
monitoring data stream	RTCP or proprietary	Streaming Messaging service
end-to-end delivery of stream	RTP or proprietary	Streaming Messaging service
message transport	UDP, Multicast UDP, TCP	Message Transport service
packet forwarding/ internetworking	IP, IP Multicast	Packet Forwarding/ Internetworking service

FIG. **20** depicts Streaming, in which a real-time data stream is transferred.

Possible Product OptionsOptions

Netscape's Media Server; Progressive Networks Real Audio/Video; VXtremes WebTheater

The following are examples of products that implement Streaming Messaging (based upon RTSP or other standards or proprietary approaches):

Netscape's Media Server

Progressive Networks' Real Video VXtreme's WebTheater

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 161 of 281

Specialized Messaging **1538**

Specialized Messaging services extend the Core Messaging services to provide additional functionality, including:

- Provides messaging among specialized systems by drawing upon basic messaging capabilities
- Defines specialized message layouts
- Defines specialized inter-system protocols
- Suggests ways in which messaging draws upon directory and security services in order to deliver a complete messaging environment

An example of a specialized messaging service is Mail Messaging. Mail Messaging is a specialized implementation of store-and-forwarding MOM (message-oriented middleware) messaging, in that Mail Messaging defines specialized, mail-related message layouts and protocols that utilize store-and-forward messaging.

E-Mail **1540**

E-Mail takes on a greater significance in the modern organization. The E-Mail system, providing it has sufficient integrity and stability, can function as a key channel through which work objects move within, and between organizations in the form of messages and electronic forms. An E-Mail server stores and forwards E-Mail messages. Although some products like Lotus Notes use proprietary protocols, the following protocols used by E-Mail Services are based on open standards:

- X.400—The X.400 message handling system standard defines a platform independent standard for store-and-forward message transfers among mail servers. X.400 is often used as a backbone e-mail service, with gateways providing interconnection with end-user systems.
- SMTP—Simple Mail Transfer Protocol (SMTP) is a UNIX/Internet standard for transferring e-mail among servers.
- MIME—Multi-Purpose Internet Mail Extensions (MIME) is a protocol that enables Internet users to exchange multimedia e-mail messages.
- POP**3**—Post Office Protocol (POP) is used to distribute e-mail from an SMTP server to the actual recipient.
- IMAP**4**—Internet Message Access Protocol, Version 4 (IMAP**4**) allows a client to access and manipulate electronic mail messages on a server. IMAP**4** permits manipulation of remote message folders, called "mailboxes", in a way that is functionally equivalent to local mailboxes. IMAP**4** also provides the capability for an off-line client to re-synchronize with the server. IMAP**4** includes standards for message handling features that allow users to download message header information and then decide which e-mail message contents to download.

Implementation Considerations

A number of E-mail servers from vendors including HP and Netscape are built around SMTP, and most proprietary protocol E-Mail servers now provide SMTP gateways.

The Multi-part Internet Mail Extensions (MIME) standard has gained acceptance as the Internet mechanism for sending E-mail containing various multimedia parts, such as images, audio files, and movies. S/MIME, or secure MIME adds encryption and enables a secure mechanism for transferring files.

Although currently POP**3** is the popular Internet E-Mail message handling protocol, recently the lesser known IMAP**4** protocol has been gaining in adoption among mail server and mail client software providers. IMAP was designed to add features beyond POP that allow users to store and archive messages and support mobile users that need to keep messages on a central server as well as on their laptop.

Organizations are looking to use vehicles like E-Mail and the Internet to enable communications with customers and trading partners. The least common denominator E-mail capability today is very rudimentary (ASCII text). But as the standards listed here as well as others become integrated into most of the popular E-mail products and gateways this will change enabling a more flexible and useful commercial communications medium.

Possible Product OptionsOptions

Microsoft Exchange Server; Lotus cc:mail; Lotus Notes; Qualcomm Eudora; TenFours TFS Universal E-Mail Gateway; UUcoding; Netscape Mail Server; Post.Office; NTMail

The following E-Mail products are based on the open Internet standards defined above:

- Netscape Mail Server—Netscapes implementation of an open standards-based client/server messaging system that lets users exchange information within a company as well as across the Internet. It includes support for all standard protocols, and is packaged with Netscapes SuiteSpot server line.
- Post.Office—one of the leading POP**3**/SMTP mail servers for the Internet community as well as corporate intranets. This message transport agent is based entirely on the open standards of the Internet, ensuring maximum compatibility with other systems.
- NTMail—an open SMTP and POP**3** mail server for Windows NT.

The following are major proprietary E-mail servers used in large organizations today:

- Lotus Notes—platform-independent client/server mail system. Notes Mail can support over 1,500 active users per server, offering Internet integration, distributed replication and synchronization. Lotus Notes also provides integrated document libraries, workflow, calendaring and scheduling, and a cc:Mail user interface.
- Microsofts Exchange Server—Exchange 4.0 provides a messaging and groupware platform to support collaboration solutions on Windows machines. Microsoft Exchange 5.0 has support for all of the key Internet protocols. These include POP**3** for mailbox access, SMTP for mail sending and receiving, NNTP for newsgroups and discussion forums, LDAP for directory access, HTTP and HTML for access via a web browser, and SSL for security.

The following products are examples of e-mail systems:

Microsoft Mail

Lotus cc:mail

Qualcomm Eudora

The following products provides e-mail system translation:

- TenFour's TFS Universal E-Mail Gateway—links users of Lotus Development Corp.'s cc:Mail and Notes, Novell Inc.'s GroupWise, Microsoft Corp.'s Mail, MCI Mail, and SMTP e-mail to Microsoft Exchange.
- UUcoding—process for converting 8-bit binary files into 7-bit ASCII files for transmission via e-mail over the Internet (the Internet only supports seven bit characters in e-mail messages); UUencode and UUdecode utilities on end nodes perform the conversion.

Database Access **1542**

Database Messaging services (also known as Database Access Middleware) provide connectivity for clients to access databases throughout the enterprise. Database messaging software draws upon basic inter-process messaging capabilities (e.g., RPCs) in order to support database con-

nectivity. Database Messaging services typically provide single application seemless access to mulitple data sources, both relational and non-relational. Additionally, database messaging services can be used to facilitate migration of data from one environment to another (i.e., MVS/DB2→Sybase)

There are three types of database access middleware:

ODBC-like

Propietary

Gateway

Is there a Projected Growth in Data Requirements?

Storage of data in a database allows for more optimal future growth since databases scale better than mechanisms such as flat files.

Should the Data be Secured and Controlled?

Use databases to protect data integrity from multiple user access, and hardware and software failures.

Is it Desirable to Limit the Amount of Viewed data?

Use databases to store large amounts of information and to access an individual record(s) without having to inspect all the records of a given topic.

Is There a Need to Impose Data Standards?

Use a database when you wish to store and impose standards on data elements. This is important when developing enterprise wide solutions, since it is desirable to have the different applications access the same structured information.

Is there a Current or Potential Requirement for a Distributed Architecture?

Databases allow for the potential of such architectural features as a data replication strategy and/or distributed data access.

Is there a Need to Minimize Data Duplication?

Because of their normalized design, relational databases are used to reduce data redundancy. This reduces maintenance and storage requirements.

Product Considerations

What are the Available Administration or Systems Management Features?

Administration and systems management features such as remote management, remote configuration, backup and recovery, and disaster recovery should be considered.

What are the key Business Requirements?

Product selection may be influenced by business requirements such as replication and distributed data, parallel processing, complex object support for such purposes as multimedia, OLTP, decision support, VLDB, data warehousing, and availability (24/7 vs. 8/5).

What is the Availability of Market Resources to Support the Product?

Personnel available for support (permanent hires, contractors), and third party support for skilled resources/training should be considered.

Are the Current Data Requirements Expected to Increase?

Products differ in their ability to scale with respect to hardware architecture, transaction throughput, and user base.

How do the Vendors Compare Against one Another?

Issues to consider are type, quality and responsiveness of support, alliances/partnerships with other companies, market presence (install base, customer list, number of production copies, etc.), vendor industry, alignment of mission and vision with that of potential customer/evaluator, product philosophy, long-term product plans/strategy, and vendor's training.

How Well Does a Product Integrate With the Current or Proposed Architecture?

Issues to consider include supported operating systems, networks, and other database platforms, availability of database utilities, application interfaces, development tools, and third party products, and integration with legacy systems.

Possible Product Options

Oracles SQL*Net; Sybases EnterpriseConnectivity; Microsoft's Open Database Connectivity (ODBC); Sun Java Database Connectivity (JDBC)

- Oracle's SQL*Net—supports database interoperability across a variety of transport protocols (e.g., TCP/IP, SPX/IPX, SNA, etc.); includes verbs such as connect, send, receive, and disconnect; performs transparent protocol bridging by allowing multiple protocols to reside simultaneously on each node.
- Sybase's EnterpriseConnectivity—supports database interoperability across a variety of platforms.
- Microsoft's Open Database Connectivity (ODBC)—a database programming interface that provides a common language for Windows applications to access databases on a network.
- Sun's Java Database Connectivity (JDBC)—a Java-based programming interface that provide a common method for Java applications to access databases on a network

Object Messaging **1544**

Object Messaging enables objects to transparently make requests of and receive responses from other objects located locally or remotely. Objects communicate through an Object Request Broker (ORB). An ORB enables client objects to access server objects either locally or remotely over a network and invoke operations (i.e. functions and methods) on them. ORBs typically provide interoperability between heterogeneous client and server environments: across languages and/or operating systems and/or network protocols. In that respect some have said that ORBs will become a kind of "ultimate middleware" for truly distributed processing. A standardized Interface Definition Language (IDL) defines the interfaces that applications must use to access the ORB Services. The two major Object Request Broker standards/implementations are:

- Object Management Group's Common Object Request Broker Architecture (CORBA)
- Microsoft's (Distributed) Component Object Model (COM/DCOM)

CORBA

Common Object Request Broker Architecture (CORBA) is a standard for distributed objects being developed by the Object Management Group (OMG). The OMG is a consortium of software vendors and end users. Many OMG member companies are developing commercial products that support the CORBA standards and/or are developing software, that use these standards. CORBA provides the mechanism by which objects transparently make requests and receive responses, as defined by OMG's Object Request Broker (ORB). The CORBA ORB is an application framework that provides interoperability between objects, built in different languages, running on different machines in heterogeneous distributed environments.

Inter-ORB Messaging

The OMGs Internet Inter-Orb Protocol (IIOP) specifies a set of message formats and common data representations for communication between ORBs over TCP/IP networks. CORBA-based Object Messaging is summarized in FIG. **21**.

COM/DCOM

Component Object Model (COM) is a client/server object-based model, developed by Microsoft, designed to

allow software components and applications to interact with each other in a uniform and standard way. The COM standard is partly a specification and partly an implementation. The specification defines mechanisms for creation of objects and communication between objects. This part of the specification is paper-based and is not dependent on any particular language or operating system. Any language can be used as long as the standard is incorporated. The implementation part is the COM library which provides a number of services that support a mechanism which allows applications to connect to each other as software objects. COM is not a software layer through which all communications between objects occur. Instead, COM serves as a broker and name space keeper to connect a client and an object, but once that connection is established, the client and object communicate directly without having the overhead of passing through a central piece of API code. Originally conceived of as a compound document architecture, COM has been evolved to a full object request broker including recently added features for distributed object computing. DCOM (Distributed COM) contains features for extending the object model across the network using the DCE Remote Procedure Call (RPC) mechanism. In sum, COM defines how components should be built and how they should interact. DCOM defines how they should be distributed. Currently COM/DCOM is only supported on Windows-based machines. However, third-party vendors are in progress of porting this object model to other platforms such as Macintosh, UNIX, etc. FIG. **22** illustrates COM Messaging.

Implementation Considerations

Although ORBs provide a mechanism for transparently communicating among components located locally or remotely, performance issues need to be thoroughly addressed before moving components around the network Making requests and receiving responses among components located on different machines will take longer that having the same communication between components located on the same machine. Performance is dependent on what type of network is available (LAN, type of LAN, WAN, type of WAN, dial-up, wireless, etc.), size of messages and number of messages that go across the network.

Possible Product Options

Expersoft's CORBAplus; IBM's Component Broker; BEASystems ObjectBroker; Iona Technology's Orbix; Inprise's Visibroker; Microsofts COM; Software AGs COM

- CORBA-based ORB Products
  - Expersoft's CORBAplus
  - IBM's Component Broker
  - BEA's Object Broker
  - Iona Technologies's Orbix
  - Inprise's VisiBroker(formerly Visigenic)
- COM Products
  - Microsoft's DCOM (Windows NT Server, Windows NT Workstation, Windows 95, Apple Macintosh, Windows Java Virtual Machine)
  - Software AG's COM (current or planned availability on Sun, Digital UNIX, IBM, and HP platforms)

CTI Messaging **1546**

Computer-Telephone Integration (CTI) integrates computer systems and telephone systems to coordinate data and telephony activities. For example, CTI can be used to associate a customers database entry with the customers telephone call and route the call accordingly.

Referring to FIG. **23**, CTI Messaging supports communication among clients **2300**, CTI servers **2302**, PBXs/ACDs **2304**, hybrid platforms, networks **2306**, and external telephony devices. CTI Messaging relies upon proprietary PBX/ACD APIs, CTI vendor-specific APIs or message sets, and industry-standard APIs.

- CTI Messaging has two primary functions:
  - Device-specific communication
    - Manages direct communications between telephony devices and data devices
    - Allows applications to control PBXs, key telephone systems, ISDN, analog PSTN, cellular, Centrex, etc. and supports features such as address translation, call setup, call answering, call dropping, and caller ID.
    - Provides interface to carrier networks for call delivery and call-related messaging
  - Message mapping
    - Translates device-specific communication to generic API and/or message set
- CTI products can be divided into the following categories:
  - CTI Platform-Specific Products—products that can only be implemented on the hardware of a specific vendor.
    - CTI Telephony-based API Products—include proprietary PBX/ACD-based messaging sets, which permit external devices to interface with the vendor's PBX/ACD call and station control logic
    - CTI Server/Workstation-based or Host-based API Products—operate on a particular computer vendor's hardware platform and provide call control and messaging functionality.
  - CTI Cross-Platform Vendors—products that have been ported to multiple hardware platforms/operating systems.
    - CTI Enabling Solutions—focus solely on call control and call/application synchronization functions.
    - CTI Enterprise Solutions—provide all CTI business functions to varying degrees.

Possible Product Options

Novell's Netware Telephony Services; Microsoft TAPI; Novell TSAPI

Industry-Standard Application Programming Interfaces (APIs):

- Microsoft's TAPI
- Novell's TSAPI
  - Novell's Netware Telephony Services—Based on Novell's Telephony Services API (TSAPI), Netware Telephony Services is a CTI gateway that integrates Novell networks with telephony networks.
- Other vendors of CTI products include:
  - Aspect Telecommunications Corp.
  - Genesys Labs
  - IBM
  - Lucent
  - Nortel
  - Rockwell

EDI Messaging **1548**

EDI (Electronic Data Interchange) supports system-to-system messaging among business partners by defining standard message layouts. Companies typically use EDI to streamline commercial transactions within their supply chains. EDI standards (e.g., EDIFACT, ANSI X**12**) define record layouts for transactions such as "purchase orders". EDI services include the generation and translation of EDI messages according to the various public message layout standards.

EDI messaging can be implemented via electronic mail or customized message-oriented architectures.

Implementation Considerations

EDI messages have traditionally been sent between companies using a VAN (Value Added Network). VANs have been criticized for their relatively high cost in comparison to public networks like the Internet. Recently, EDI messaging vendors such as Premenos have been creating software with built-in encryption features to enable companies to send EDI transmissions securely over the Internet.

Web server vendors including Microsoft, Netscape and OpenMarket are putting plans in place to add EDI transmission capabilities into their Web server products. OpenMarket Inc. is working with Sterling and Premenos to integrate their EDI management software with OpenMarkets OMTransact electronic commerce server software. Netscape is working with GEIS in creating Actra Business Systems to integrate EDI services with Netscape server products.

Possible Product Options

Digital Equipment Corp.s DEC/EDI; Sterling Commerces GENTRAN; IBM Global Services Advantis; GE Information Services; Sterling Commerce

EDI Applications

Digital Equipment Corp.'s DEC/EDI

Sterling Commerce's GENTRAN

EDI value-added networks (VANS)—VANs link EDI trading partners and transmit EDI messages through a central electronic clearinghouse

IBM Global Services' Advantis

GE Information Services

Sterling Commerce

Legacy Integration **1550**

Legacy services provide gatewarys to mainframe legacy systems. The following protocol is typically used:

Systems Network Architecture (SNA) is a networking connection-oriented protocol architecture which was developed in the 1970s by IBM. Currently, SNA and TCP/IP are two of the most widely used networking protocol architectures.

Design techniques for integration with existing systems can be grouped into two broad categories:

Front end access—discussed as part of Terminal Emulation

Back end access—tend to be used when existing data stores have information that is needed in the client/server environment but accessing the information through existing screens or functions is not feasible. Legacy Integration messaging services typically include remote data access through gateways. A database gateway provides an interface between the client/server environment and the legacy system. The gateway provides an ability to access and manipulate the data in the legacy system.

Implementation Considerations

Legacy systems hold critical data which must be accessible by new Netcentric computing solutions. These legacy data sources often must be accessed in their current form so as to not disrupt the legacy systems.

Communications Security **1508**

Communications Security services control access to network-attached resources. Combining network Security services with security services in other parts of the system architecture (e.g., application and database layers) results in robust security.

Possible Product Options

UkWeb's Stronghold; UkWeb's SafePassage

UkWeb's Stronghold

Stronghold was the first web server to support SSL Client Authentication. Regular expression-based matching of client certificate information to determine access control is possible. Stronghold also has an API for certificate to username mapping so that client certificates may be mapped to standard usemames. CA certificates from both Thawte and Verisign can be utilized. Uncompromised, full 128-bit symmetric encryption is provided in all versions. This provides Netcentric systems used outside of the USA or Canada with secure encryption capabilities.

UkWebs's SafePassage

SafePassage is a full-strength, encrypting Web proxy. It is designed to supplement the security of browsers whose authentication and encryption capabilities have been weakened to comply with United States export regulations. For these types of browsers, SafePassage will provide client authentication certificates and full-strength encryption (128 bit).

Encryption **1552**

Encryption services encrypt data prior to network transfer to prevent unauthorized interception. (Note that encryption can occur within the Communications Services layer, the Transport Services layer, or the Network Media Services layer.) Within the Communications Services layer, encryption occurs at the top of the protocol stack and is typically performed within an application (e.g., an e-mail application, a Web browser). This is an end-to-end approach that can leave the remainder of the protocol stack (i.e., the Transport services and the Network Media services) unaffected.

Encryption has two main components: the encryption algorithm, which is the series of steps that is performed to transform the original data; and the key, which is used by the algorithm in some way to encrypt the message. Typically, the algorithm is widely known, while the key is kept secret. There are several types of encryption in use today, including:

Secret key cryptography—uses one key (the secret key) both to encrypt the message on one side and to decrypt the message on the other side.

Public key cryptography—uses two keys, the public key and the private key. The public key and private key are mathematically related so that a message encrypted with the recipient's public key may be decrypted with the recipient's private key. Therefore, the public key can be widely published, while the private key is kept secret.

There are also varying methods of employing encryption types described above to encrypt data sent across a network:

Data link layer—data is encrypted before it is placed on the wire. Data link encryptors are generally hardware products.

Application layer—data is encrypted by the application. Netscape's Secure Sockets Layer (SSL) is one example of application-layer encryption for WWW browsers. SSL uses RSA encryption to wrap security information around TCP/IP based protocols.

Network layer—data is encrypted inside the network layer header, therefore relying on the network layer protocol.

Implementation Considerations

The advantage of SSL over S/HTTP is that SSL is not restricted to HTTP but can also be used for securing other TCP/IP based services such as FTP, Telnet, etc. SSL can provide session level data encryption and authentication to enable secure data communications over public networks such as the Internet.

The need for Encryption Services is particularly strong where electronic commerce solutions that involve exchanging sensitive or financial data are to be deployed over public

networks such as the Internet. Cryptography can be used to achieve secure communications, even when the transmission media (for example, the Internet) is untrustworthy. Encryption Services can also be used to encrypt data to be stored (e.g., sensitive product information on a sales person's laptop) to decrease the chance of information theft.

There are complex legal issues surrounding the use of encrypting in an international environment. The US government restricts what can be exported (in terms of encryption technology), and the French government defines encryption technology as a "weapon of war" with appropriate legal and regulatory restrictions. This is a key issue in international. e-commerce today.

Possible Product Options

Netscape's Secure Sockets Layer (SSL); S-HTTP; e-mail encryption; S-MIME

Encryption that is architected into Web-based solutions

Netscape's Secure Sockets Layer (SSL)—provides encryption for World Wide Web browsers.

S-HTTP—a secure version of the HTTP data transfer standard; used in conjunction with the World Wide Web.

Encryption that is embedded in e-mail products

e-mail encryption—products such as Lotus Notes and Microsoft Exchange can encrypt e-mail messages and/or attachments.

S-MIME—a secure version of the MIME e-mail standard.

Authorization **1554**

When a user requests access to network resources, the Authorization service determines if the user has the appropriate permissions and either allows or disallows the access. (This occurs after the user has been properly authenticated.)

The following are examples of ways to implement Authorization services:

Network Operating Systems—Authorization services are bundled with all network operating systems in order to control user access to network resources.

Firewall Services protect sensitive resources and information attached to an Intxxnet network from unauthorized access by enforcing an access control policy. A variety of mechanisms exist for protecting private networks including:

Filters—World Wide Web filters can prevent users from accessing specified content or Internet addresses. Products can limit access based on keywords, network addresses, time-of-day, user categories, etc.

Application Proxies—An application-level proxy, or application-level gateway, is a robust type of firewall. (A firewall is a system that enforces an access control policy between a trusted internal network and an untrusted external network.) The application proxy acts at the application level, rather than the network level. The proxy acts as a go-between for the end-user by completing the user-requested tasks on its own and then transferring the information to the user. The proxy manages a database of allowed user actions, which it checks prior to performing the request.

Servers, Applications, and Databases—Authorization can occur locally on a server to limit access to specific system resources or files. Applications and databases can also authorize users for specific levels of access within their control. (This functionality is within the Environment Services grouping in the execution architecture.)

Possible Product Options

Microsoft Windows NT; Novell Netware; UNIX; Check Points Firewall-1; Raptor Systems Eagle Firewall; Microsoft Proxy Server; Netscape Proxy Server; Microsystem Softwares Cyber Patrol Corporate; Net Nanny Softwares Net Nanny

Network Operating Systems

Microsoft Windows NT, Novell Netware, UNIX, etc.

application Proxies

Microsoft Proxy Server—allows for designation of who can access the Internet and which services they can use. Administrators can establish additional credentials for logging on, set specific dialing hours or days of the week, and restrict access to certain sites altogether.

Netscape Proxy Server—high-performance server software for replicating and filtering access to Web content on the Internet or an intranet. Provides access control, URL filtering, and virus scanning.

Filters

Check Point FireWall-1—combines Internet, intranet and remote user access control with strong authentication, encryption and network address translation (NAT) services. The product is transparent to network users and supports multiple protocols.

BorderWare Firewall—protects TCP/IP networks from unwanted external access as well as provides control of internal access to external services; supports packet filters and application-level proxies.

Raptor System's Eagle Firewall

Microsystem Software's Cyber Patrol Corporate

Net Nanny Software's Net Nanny

Authentication

Authentication services verify network access requests by validating that users are who they claim to be. For secure systems, one or more authentication mechanisms can be used to validate authorized users and to verify which functions and data they have access to. Within the corporate network, authentication services are often included in directory services products like Novell's NDS. NDS requires the user to have an established account and supply a password before access is granted to resources through the directory.

Authentication for accessing resources across an Internet or intranet is not as simple and is a rapidly evolving area. When building e-commerce Web sites there may be a need to restrict access to areas of information and functionality to known customers or trading partners. More granular authentication is required where sensitive individual customer account information must be protected from other customers.

Authentication can occur through various means:

Basic Authentication—requires that the Web client supply a user name and password before servicing a request. Basic Authentication does not encrypt the password in any way, and thus the password travels in the clear over the network where it could be detected with a network sniffer program or device. Basic authentication is not secure enough for banking applications or anywhere where there may be a financial incentive for someone to steal someone's account information. Basic authentication is however the easiest mechanism to setup and administer and requires no special software at the Web client.

ID/Password Encryption—offers a somewhat higher level of security by requiring that the user name and password be encrypted during transit. The user name and password are transmitted as a scrambled message as

part of each request because there is no persistent connection open between the Web client and the; Web server.

Digital Certificates or Signatures—encrypted digital keys that are issued by a third party "trusted" organization (i.e. Verisign); used to verify user's authenticity.

Hardware tokens—small physical devices that may generate a one-time password or that may be inserted into a card reader for authentication purposes.

Virtual tokens—typically a file on a floppy or hard drive used for authentication (e.g. Lotus Notes ID file).

Biometric identification—the analysis of biological characteristics to verify individuals identify (e.g., fingerprints, voice recognition, retinal scans).

Related to authentication, non-repudiation is a means of tagging a message in order to prevent an entity from denying that it sent or received the message.

Possible Product Options

Microsoft Windows NT; Novell NetWare; UNIX; Platinum Technologies AutoSecure SSO; Axents Enterprise Access Control for Windows 95; SecurID; Racals TrustMe Authentication Server; Visionics FaceIt; Sensars IrisIdent; Keyware Technologies Voice Guardian; National Registrys NRIdentity; Kerberos; VeriSign

The following are examples of products that perform authentication:

user IDs and passwords

operating systems: Microsoft Windows NT, Novell NetWare, UNIX, etc.

application level user IDs and passwords (e.g., e-mail system) single sign-on software—manages user logins to multiple systems or resources.

Platinum Technologies' AutoSecure SSO add-on administration packages—enhance the capabilities of native operating system security

Axent's Enterprise Access Control for Windows 95—enforces password standards and encrypts data.

Hardware Tokens

Security Dynamics' SecurID Authentication Tokens

Racal's TrustMe Authentication Server

Biometric Security

Visionics' FaceIt—face recognition

Sensar's Irisldent—iris identification

Keyware Technologies' Voice Guardian—voice recognition

National Registry's NRIdentity—fingerprint recognition keys and certificates

Kerberos—an encryption and key management protocol for third party authorization; vendors include CyberSAFE and Digital Equipment Corporation.

VeriSign—a company that manages digital certificates.

COMMUNICATION FABRIC **1010**

As communication networks become increasingly complicated and interconnected, the services provided by the network itself have by necessity increased as well. Clients and servers are rarely directly connected to one another, but separated by a network of routers, servers and firewalls providing an ever increasing number of network services such as address resolution, message routing, security screening and many more.

The communications fabric provides common network services to the platform-specific network services residing on the client and server nodes. These common network services can be used to manage resources and translate capabilities within and across enterprises.

Short of interpreting the data being transmitted, the communications fabric is aware of the different message-oriented information streams in order to help the client and server communicate regardless of the different network functions implemented on each platform.

An intelligent communications fabric monitors and routes data flows and provides functionality (security, directories, etc.) to clients and servers. An intelligent communications fabric provides the following benefits:

An intelligent network can manage itself, including addressing, routing, security, recovery, etc. It is inefficient for individual clients and servers to perform such tasks.

Specialized network components reduce the network-related processing that occurs on clients and servers.

An intelligent network integrates heterogeneous clients, servers, and other resources by resolving incompatible protocols and standards.

An intelligent network has the capability to actively manage the flow of information rather than simply moving data. This allows the network to effectively deliver multimedia and other network-sensitive traffic.

An intelligent network adds value to enterprise resources by presenting a cohesive view of available resources and increasing the level of security associated with those resources.

FIG. **24** illustrates various components of the Communication Fabric.

Transport Services **2402**

Provides the underlying protocols responsible for transmitting and securing data communications. Transport Services are responsible for establishing, maintaining and terminating end-to-end communications between users and processes. Connection management provides transfer services that ensure the delivery of data from sender to receiver, which support the transferring of messages from a process running on one machine to a process running on another machine. In addition, connection management provides services that initiate a connection, gracefully terminate a connection, and handle abrupt termination. These services take place for application before and after the data is formatted for transport over the network.

Messaging Transport **2404**

The Message Transport service is responsible for the end-to-end delivery of messages. It can include the following functionality:

End-to-End Data Transfer—The Message Transport Service formats messages for sending and confirms the integrity of received messages.

Connection Control—The Message Transport service may establish end-to-end (client-server) connections and track addresses and other associated information for the connection. The service also tears down connections and handles hard connection failures.

Reliable Transfer—The Message Transport service may manage reliable delivery of messages through the use of acknowledgments and retransmissions.

Flow Control—The Message Transport service may allow the receiver to govern the rate at which the sender transfers data.

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 167 of 281

Multiplexing—The Message Transport service may define multiple addresses or ports within a single network node, allowing multiple processes on the node to have their own communications paths.

(Some transport services do not implement all of the listed functionality. For example, the UDP protocol does not offer connection control or reliable transfer.)

The following are examples of protocols that provide message transport:

SPX (Sequenced Packet exchange)

TCP (Transmission Control Protocol)

UDP (User Datagram Protocol)

NetBIOS/NetBEUI (Network Basic Input/Output System/NetBIOS Extended User Interface)

APPC (Advanced Program-to-Program Communications)

AppleTalk

Packet Forwarding/Internetworking **2406**

The Packet Forwarding/Internetworking service transfers data packets and manages the path that data takes through the network. It includes the following functionality:

Fragmentation/Reassembly—The Packet Forwarding/Internetworking service divides an application message into multiple packets of a size suitable for network transmission. The individual packets include information to allow the receiving node to reassemble them into the message. The service also validates the integrity of received packets and buffers, reorders, and reassembles packets into a complete message.

Addressing—The Packet Forwarding/Internetworking service encapsulates packets with addressing information.

Routing—The Packet Forwarding/Internetworking service can maintain routing information (a view of the network topology) that is used to determine the best route for each packet. Routing decisions are made based on the cost, percent utilization, delay, reliability, and similar factors for each possible route through the network.

Switching—Switching is the process of receiving a packet, selecting an appropriate outgoing path, and sending the packet. Switching is performed by routers and switches within the communications fabric. Switching can be implemented in the following ways:

For some network protocols (e.g., IP), routers draw upon dynamic routing information to switch packets to the appropriate path. This capability is especially important when connecting independent networks or subnets.

For other network protocols (e.g., Ethernet, Token Ring), switching simply directs packets according to a table of physical addresses. The switch can build the table by "listening" to network traffic and determining which network nodes are connected to which switch port.

Some protocols such as Frame Relay involve defining permanent routes (permanent virtual circuits PVCs) within the network. Since Frame Relay is switched based upon PVCs, routing functionality is not required.

Multicasting—The Packet Forwarding/Internetworking service may support multicasting, which is the process of transferring a single message to multiple recipients at the same time. Multicasting allows a sender to transfer a single copy of the message to the communications fabric, which then distributes the message to multiple recipients.

The following are examples of protocols that provide Packet

Forwarding/Internetworking:

IP (Internet Protocol)

IP Multicast (emerging standard that uses a special range of IP addresses to instruct network routers to deliver each packet to all users involved in a multicast session)

IPX (Internetwork Packet Exchange)

ATM (Asynchronous Transfer Mode)

Frame Relay

X.25

SMDS (Switched Multimegabit Data Service)

The Following are Examples of Network Components that Perform Packet

Forwarding/Internetworking:

routers

switches

ATM switches, Frame Relay switches, IP switches, Ethernet switches, Token Ring switches, etc.

The following are examples of protocols that maintain routing information tables within routers:

Distance Vector Protocols—each router periodically informs neighboring routers as to the contents of routing table (destination addresses and routing metrics); routing decisions based on the total distance and other "costs" for each path.

IP and IPX Routing Information Protocols (RIP)

AppleTalk Routing Table Management Protocol (RTMP)

Ciscos Interior Gateway Routing Protocol (IGRP) and Enhanced IGRP

Link-State Protocols—each router periodically broadcasts changes to the routers and directly attached networks that it can talk with.

Open Shortest Path First (OSPF)

ISOs Intermediate System to Intermediate System (IS-IS)

Novells NetWare Link Services Protocol (NLSP)

Policy Routing Protocols—allow Internet backbone routers to accept routing information from neighboring backbone providers on the basis of contracts or other non-technical criteria; routing algorithms are Distance Vector.

Border Gateway Protocol (BGR)

Interdomain Routing Protocol (IDR)

Circuit Switching **2408**

While Message Transport services and Packet Forwarding/Internetworking services support the transfer of packetized data, Circuit Switching services establish physical circuits for the transfer of circuit-switched voice, fax, video, etc.

Circuit Switching

uses an end-to-end physical connection between the sender and the receiver that lasts for the duration of the "call"

includes voice, video, fax, etc.

includes data in a circuit switched architecture (e.g., dial-up connections)

Packetized

transferred through brief, temporary, logical connections between nodes

includes data and packetized multimedia (video, voice, fax, etc.)

Circuit Switching includes the following functionality:

establishes end-to-end path for circuit (may involved multiple intermediate nodes/switches)

manages end-to-end path (quality, billing, termination, etc.)

The following are examples of Circuit Switching:

analog dial-up telephone circuit

ISDN (Integrated Services Digital Network)

Possible Product Options

Lucent's Definity; Nortels Meridian; Lucents E5S; Nortels DMS; Tellabs Titan products; Lucents DSX products; Alcatels SX products; AltiGens AltiServ; Lucents Internet Telephony Server

The following are examples of PBX products, which perform circuit switching within private telephone networks:

Lucent's Definity

Nortel's Meridian

The following are examples of central office (telephone company) switches, which perform circuit switching within the public telephone network:

Lucent's E5S

Nortel's DMS

The following are examples of Digital Access Cross-connect Systems (DACS), which are configured to switch circuits among multiple ports.

Tellabs' Titan products

Lucent's DSX products

Alcatel's SX products

The following is an example of a PC-based PBX system:

AltiGen's AltiServ—PC-based PBX system for a small branch office or a low-volume specialized call center.

The following is an example of a circuit-switching/packet-forwarding gateway:

Lucent's Internet Telephony Server—server software that routes calls from PBXs over the Internet or intranets.

Transport Security **2410**

Transport Security services (within the Transport Services layer) perform encryption and filtering.

Transport-layer Encryption

Encryption within the Transport Services layer is performed by encrypting the packets generated by higher level services (e.g., Message Transport) and encapsulating them in lower level packets (e.g., Packet Forwarding/Internetworking). (Note that encryption can also occur within the Communications Services layer or the Network Media layer.) Encryption within the Transport Services layer has the advantage of being independent of both the application and the transmission media, but it may make network monitoring and troubleshooting activities more difficult.

The following standards support transport-layer encryption:

Point to Point Tunneling Protocol

Layer **2** Tunneling Protocol

Transport-layer Filtering

Network traffic can be controlled at the Transport Services layer by filtering data packets based on source and/or destination addresses and network service. This ensures that only authorized data transfers can occur. This filtering is one of the roles of a packet filtering firewall. (A firewall is a system that enforces an access control policy between a trusted internal network and an untrusted external network.)

The following IETF standard supports interoperability among security systems:

IPSec Allows two nodes to dynamically agree on a security association based on keys, encryption, authentication algorithms, and other parameters for the connection before any communications take place; operates in the IP layer and supports TCP or UDP. IPSec will be included as part of IPng, or the next generation of IP.

Implementation Considerations

Firewalls can also provide a single point of access to the companys network, which could be used to maintain an audit trail. Some firewalls provide summaries to the administrator about the type of traffic and amount of traffic passed through it, number of break in attempts, etc.

Most commercial firewalls are configured to reject all network traffic that has not been explicitly allowed, thus enforcing the policy, Only allow traffic that has been categorically permitted, otherwise prohibit. This policy provides much more control and is much safer than a policy which allows traffic unless it has been explicitly prohibited.

Possible Product Options

Cisco Systems; Bay Networks; 3Com Corp.; Check Points Firewall-1; Raptor Systems Eagle Firewall; Data Fellows F-Secure VPN; Racals Datacryptor 64F

The following are examples of vendors of products that perform Transport-level encryption:

routers:
- Cisco Systems
- Bay Networks
- 3Com Corp.

firewalls:
- Check Point's Firewall-1
- Secure Computing's BorderWare Firewall Server
- Raptor Systems' Eagle Firewall

encryption devices:
- Data Fellows' F-Secure VPN
- Racal's Datacryptor 64F

The following are examples of products that perform Transport-level packet filtering:

firewalls:
- Check Point FireWall-1—combines Internet, intranet and remote user access control with strong authentication, encryption and network address translation (NAT) services. The product is transparent to network users and supports multiple protocols.
- Secure Computing's BorderWare Firewall Server protects TCP/IP networks from unwanted external access as well as provides control of internal access to external services; supports packet filters and application-level proxies.
- Raptor Systems' Eagle Firewall

routers:
- Cisco Systems
- Bay Networks
- 3Com Corp.

Network Address Allocation **2412**

Network Address Allocation services manage the distribution of addresses to network nodes. This provides more flexibility compared to having all nodes assigned static addresses. This service assigns addresses to nodes when they initially power-on and connect to the network.

The following are examples of standards that implement Network Address Allocation and allow a network node to ask a central resource for the node_s network address (e.g., IP address):

DHCP (Dynamic Host Configuration Protocol)

BootP (Bootstrap Protocol)

Quality of service **2414**

Different types of network traffic (e.g., data, voice, video) have different quality of service requirements. For example, data associated with video conferencing sessions is useless

if it is not delivered "on time". On the other hand, traditional best-effort data services, such as file or e-mail transfer, are not affected by variations in latency. QoS (Quality of Service) services deliver a defined network throughput for designated traffic by allocating dedicated bandwidth, prioritizing data traffic, etc. (Note that as an alternative to predefined throughput, some QoS protocols can also offer a best effort (i.e., variable) throughput QoS based on available network capacity.)

The following list provides a description of various Quality of Service parameters.

connection establishment delay—time between the connection request and a confirm being received by the requester

connection establishment failure probability—chance that the connection will not be established within the maximum establishment delay

throughput—bits per second (bps) of transmitted data transit delay—time elapsed between when sender transfers packet and recipient receives packet residual error rate—number of lost or corrupted messages compared to total messages in the sampling period

transfer failure probability—the fraction of the time when the throughput, transit delay, or residual error were not those agreed upon at the start of the connection

connection release delay—time between when one node initiates a release and the other node performs the release

connection release failure probability—fraction of release attempts which do not succeed

protection—specifies a secure connection

priority—indicates traffic priority over the connection

resilience—probability that the transport layer spontaneously terminates QoS can be achieved in various ways as listed below:

Specialized QoS Communications Protocols—provide guaranteed QoS.

Asynchronous Transfer Mode (ATM)—ATM is a connection-oriented wide area and local area networking protocol that delivers QoS on a per-connection basis. QoS is negotiated as part of the initial connection set up and as network conditions change. Because of the small size of ATM data cells, QoS can be better managed, compared to protocols such as Ethernet that have large frames that can tie up network components. For ATM to deliver QOS to applications, ATM must be used end-to-end.

Resource Reservation Protocol (RSVP)—The emerging RSVP specification, proposed by the Internet Engineering Task Force (IETF), allows applications to reserve router bandwidth for delay-sensitive IP traffic. With RSVP, QoS is negotiated for each application connection. RSVP enables the network to reserve resources from end to end, using Frame Relay techniques on Frame Relay networks, ATM techniques on ATM, and so on. In this way, RSVP can achieve QoS across a variety of network technologies, as long as all intermediate nodes are RSVP-capable.

IP Stream Switching—improves network performance but does not guarantee QoS.

IP Switching—IP Switching is an emerging technology that can increase network throughput for streams of data by combining IP routing software with ATM switching hardware. With IP Switching, an IP switch analyzes each stream of packets directed from a single source to a specific destination, and classifies it as short- or long-lived. Long-lived flows are assigned ATM Virtual Channels (VCs) that bypass the IP router and move through the switching fabric at the full ATM line speed. Short-lived flows continue to be routed through traditional store-and-forward transfer.

Tag Switching—Like IP Switching, emerging Tag Switching technology also improves network throughput for IP data streams. Tag Switching aggregates one or more data streams destined for the same location and assigns a single tag to all associated packets. This allows routers to more efficiently transfer the tagged data. Tag Switching is also known as Multiprotocol Label Switching.

Data Prioritization—improves network performance but does not guarantee QoS.

While not an example of end-to-end QoS, various network components can be configured to prioritize their handling of specified types of traffic. For example, routers can be configured to handle legacy mainframe traffic (SNA) in front of other traffic (e.g., TCP/IP). A similar technique is the use of prioritized circuits within Frame Relay, in which the Frame Relay network vendor assigns different priorities to different permanent virtual circuits.

Prioritization techniques are of limited effectiveness if data must also pass through network components that are not configured for prioritization (e.g., network components run by third party network providers).

Network Media Services **2416**

The Network Media layer provides the following capabilities:

Final framing of data for interfacing with the physical network.

Performing, receiving, interpreting and acting on signals from the communications fabric.

Transferring data through the physical network.

The technologies used at the Network Media Services layer vary depending on the type of network under consideration.

Media Access **2418**

Media Access services manage the low-level transfer of data between network nodes. Media Access services perform the following functions:

Physical Addressing—The Media Access service encapsulates packets with physical address information used by the data link protocol (e.g., Ethernet, Frame Relay).

Packet Transfer—The Media Access service uses the data link communications protocol to frame packets and transfer them to another computer on the same network/subnetwork.

Shared Access—The Media Access service provides a method for multiple network nodes to share access to a physical network. Shared Access schemes include the following:

CSMA/CD—Carrier Sense Multiple Access with Collision Detection. A method by which multiple nodes can access a shared physical media by "listening" until no other transmissions are detected and then transmitting and checking to see if simultaneous transmission occurred.

token passing—A method of managing access to a shared physical media by circulating a token (a special control message) among nodes to designate which node has the right to transmit.

multiplexing—A method of sharing physical media among nodes by consolidating multiple, independent channels into a single circuit. The independent channels (assigned to nodes, applications, or voice calls) can be combined in the following ways:

time division multiplexing (TDM)—use of a circuit is divided into a series of time slots, and each independent channel is assigned its own periodic slot.

frequency division multiplexing (FDM)—each independent channel is assigned its own frequency range, allowing all channels to be carried simultaneously.

Flow Control—The Media Access service manages the flow of data to account for differing data transfer rates between devices. For example, flow control would have to limit outbound traffic if a receiving machine or intermediate node operates at a slower data rate, possibly due to the use of different network technologies and topologies or due to excess network traffic at a node.

Error Recovery—The Media Access service performs error recovery, which is the capability to detect and possibly resolve data corruption that occurs during transmission. Error recovery involves the use of checksums, parity bits, etc.

Encryption—The Media Access service may perform encryption. (Note that encryption can also occur within the Communications Services layer or the Transport Services layer.) Within the Network Media Services layer, encryption occurs as part of the data link protocol (e.g. Ethernet, frame relay). In this case, all data is encrypted before it is placed on the wire. Such encryption tools are generally hardware products. Encryption at this level has the advantage of being transparent to higher level services. However, because it is dependent on the data link protocol, it has the disadvantage of requiring a different solution for each data link protocol.

The following are examples of Media Access protocols:

Ethernet

token ring

FDDI

portions of the ATM standard

HDLC/SDLC

LAPB

T-carrier, E-carrier (e.g., T**1**, T**3**, E**1**, E**3**)

TDM and FDM (Time Division Multiplexing and Frequency Division Multiplexing; used on T-carriers, etc.)

SONET, SDH

PPP, SLIP

V.**32**, V.**34**, V.**34** bis, etc.

RS-**232**, EIA-**232**

TDMA and FDMA (Time Division Multiple Access and Frequency Division Multiple Access; used on wireless links) Specialized services convert between addresses at the Media Access level (i.e., physical addresses like Ethernet) and the Packet Forwarding/Internetworking level (i.e., network addresses like IP). The following protocols are examples of this functionality:

ARP (Address Resolution Protocol)—allows a node to obtain the physical address for another node when only the IP address is known.

RARP (Reverse Address Resolution Protocol)—allows a node to obtain the IP address for another node when only the physical address is known.

Possible Product Options

Semaphores Network Security System for Workgroups

Semaphore's Network Security System for Workgroups—encrypts Ethernet.

Physical Media **2420**

As illustrated in FIG. **25**, the Physical Media is divided into two categories:

1). the physical connectors **2502**

2). the physical media (wired or wireless) **2504**

Physical Connectors

The following are examples of wiring connectors used to connect network nodes to physical media:

RJ-**11**, RJ-**45**

BNC

DB-**9**, DB-**25**

fiber optic connectors

Physical Media

Physical Media may be wired or wireless. Wired Physical Media includes wiring and cabling, while wireless Physical Media includes antennas, connectors, and the radio frequency spectrum.

The following are examples of wired physical media:

twisted pair wiring, shielded twisted pair wiring

coaxial cable

fiber optic cable

4-pair voice-grade wiring

The following are examples of wireless physical media:

cellular antennas and the associated radio frequencies

wireless local area network antennas and the associated radio frequencies

satellite antennas and the associated radio frequencies

TRANSACTION **1012**,**1014**

A transaction is a unit of work that has the following (ACID)) characteristics:

A transaction is atomic; if interrupted by failure, all effects are undone (rolled back).

A transaction produces consistent results; the effects of a transaction preserve invariant properties.

A transaction is isolated; its intermediate states are not visible to other transactions. Transactions appear to execute serially, even if they are performed concurrently.

A transaction is durable; the effects of a completed transaction are persistent; they are never lost (except in a catastrophic failure).

A transaction can be terminated in one of two ways: the transaction is either committed or rolled back. When a transaction is committed, all changes made by the associated requests are made permanent. When a transaction is rolled back, all changes made by the associated requests are undone.

Transaction Services provide the transaction integrity mechanism for the application. This allows all data activities within a single business event to be grouped as a single, logical unit of work.

In small to moderate scale environments of less than 150 simultaneous users on a single server, this service may be provided by the DBMS software with its re-start/recovery and integrity capabilities.

For larger client/server environments distributed on-line transaction managers might be more applicable. These transaction managers provide sharing of server processes across a large community of users and can be more efficient than the DBMSs.

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 171 of 281

FIG. **26** illustrates several of the components of the Transaction areas of the Netcentric Architecture Framework, each of which will be discussed in more detail below.

Transaction Monitor **2602**

The Transaction Monitor Services are the primary interface through which applications invoke Transaction Services and receive status and error information. Transaction Monitor Services, in conjunction with Information Access and Communication Services provide for load balancing across processors or machines and location transparency for distributed transaction processing.

Implementation Considerations

Does the System Access Nonrelational Data?

Some TP monitors provide a method of accessing nonrelational data, such as VSAM files or flat files, independently of where the file physically resides. If write access is required for these data sources, a TP monitor would provide more dependable messaging and two-phase commit capabilities than the data source messaging capabilities alone.

Does the System Require High Throughput?

Because TP monitors provide load balancing functionality and because they effectively reduce the number of connections that must be made to the database(s), they will help conserve the resources of the data servers and, as a result, increase the throughput of the system. Systems with high throughput requirements should consider using a TP monitor.

Do the On-line Applications Need the Support of Interoperability Between Autonomous, Heterogeneous Processors?

Some TP monitors are available on multiple platforms and maintain interoperability (communication, data translation, etc.) between heterogeneous resource managers (databases) and clients (UNIX, MS Windows NT, etc.). For this reason, projects that intend to support a heterogeneous hardware environment should consider using a TP monitor.

Is the System Supposed to be Highly Available (i.e. 24×7), or Mission Critical?

TP monitors offer robust functionality: two-phase commit, recovery/rollback, naming services, security services, can guarantee message delivery, can be maintained for high-availability operation and provides audit trail logging. Therefore, the more mission critical the system, the more likely it is that a TP monitor should be used.

Does the System Require High Availability?

Because of their fault tolerance, TP monitors make a valuable addition to systems that require high availability. The automatic restart/recovery feature helps a system recognize when components have failed and attempts to restart them. Also, because of the location transparency feature of service calling, if an entire node in a system goes down, clients may be able to reach the service they need on another node providing the same service.

Will the System be Scaled in the Future?

TP monitors offer multiple scalability options. TP monitors can run on machines ranging from PCs to mainframes. Monitors also scale by allowing new machines to be added dynamically to the system. Adding additional nodes in the production cycle is one TP monitor strength, although some monitors are better at doing this than others. If it is anticipated that system volumes will increase during the system's lifetime, scalability in itself is an excellent reason to use a TP monitor.

Does the System Have Complex Security Requirements?

All of the TP monitors available today provide security authorization/authentication services. Most of them utilize the Kerberos security package, developed at the Massachusetts Institute of Technology (MIT).

Does the System Access Legacy Systems?

TP monitors can access databases and services running on mainframe systems. TP monitors frequently include mainframe networking capability and maintain transaction rollback during mainframe accesses. If access to the legacy system is read only, the messaging capabilities of the data source will probably be sufficient. If access is write, however, the messaging and two-phase commit capabilities of the TP monitor would be more dependable.

Is the System Distributed Across Multiple Nodes?

TP monitors provide common administrative facilities to manage groups of servers. These facilities allow a system to be managed from one location with a common set of commands for each machine.

How Many Users Access the System Concurrently?

Different sources give different answers as to the number of concurrent users that necessitates the use of a TP monitor. The monitor vendors themselves give low values; the database vendors give high values. The middle ground seems to be somewhere around 250 users. This is by no means definitive, however; weigh each of the following questions when making the choice.

Do the On-line Applications Access/update More Than one Database or More Than One Type of Database?

The real strength of TP monitors is their ability to ensure a global two-phase commit over multiple, heterogeneous databases. A system that has this quality is a candidate for a TP monitor.

Is the System not a Transaction Processing System?

Although TP monitors provide global two-phase commit "transaction processing"functionality, systems that do not need this feature can also benefit by using TP monitors. For example, the load-balancing feature in itself can help increase system performance. Also, the administrative facilities can help simplify system management.

Is Data Dependent Routing Necessary?

Data Dependent Routing is the ability to route requests to a particular server based upon the data passed, within the request. TP monitors can provide this functionality. e.g. A system has several servers accepting requests from clients dispersed across North America. There are two groups of servers. One group of servers handles requests from all clients located in the USA while the other group serves requests from Canada. When a client sends a request to the system, a field in the request message, defining the location of the client, is passed to the system. The TP monitor is then able to route the request to the correct group of servers (USA or Canada) based upon information in the request message.

Is Reliable Queueing Necessary?

TP monitors provide the ability to enqueue and dequeue requests to and from a reliable

(stable storage) queue. Both the application and the administrator can control the order of the messages (service requests) in the queue. Messages can be ordered LIFO, FIFO, time based, priority, or by some combination of these keys.

Example

A system updates a customer database. Suppose that the database has been partitioned such that the information on customers most likely to use a branch office is stored locally at a branch office. There is a requirement to maintain an up-to-date of the entire customer database at the home office. The application that updates the local customer master can place a copy of the update into a reliable queue. The queue can be forwarded to the home office via a WAN, and the updates can be replicated in the home office database. The

queuing system can be used to assure that every update completed at the local office is completed at the home office.

Is The System Multi-tiered?

Transaction Services are typically used in three-tier and multi-tier architectures. Particularly in Netcentric environments, applications might need to support getting and providing access to multiple back-end services, across enterprises, as part of a single transaction or user activity. This can be especially challenging if the user does not own some or all of the back-end services and/or data that its application relies on.

Product Considerations

Is the Client Interested in Stable or Emerging Technologies?

TUXEDO has been in the TP marketplace for seven years and has the most installations of all TP monitors. Encina, TOP END, and CICS/6000 are relatively new and emerging.

Does the Client Plan to use Windows NT?

On Which Platforms/operating Systems do the Servers run?

TP monitor support for NT may be limited.

Some TP monitors are capable of running on a wider variety of platforms/operating systems than others.

Is the Project Installing a new System or Rehosting/downsizing an Existing Mainframe System?

The UniKix, VIS/TP, and CICS/6000 monitors were developed specifically with rehosting in mind. TUXEDO, Encina, and TOP END are best suited to fresh installations.

Does the System use PC-based Clients?

Each TP monitor offers different support for PC-based clients. TUXEDO and TOP END currently provide DOS, Windows, and OS/2 application programming interface (API) support. Encina offers PC support, but this feature is still in beta test. Several vendors have PowerBuilder and Visual Basic interfaces planned for their monitors, but as of this practice aid's printing, nothing has been released.

On Which Platforms Will Client Applications Execute?

New and re-engineered systems may be required to execute on a previously installed base of clients.

Does the System Require Integration With Other 3rd Party Tools?

The client may expect the TP monitor to integrate with an already installed base of 3rd party development tools.

Does the System Require Mainframe Connectivity?

Of the four monitors that are evaluated in this practice aid, all of them offer varying levels of mainframe connectivity.

Does the Client Have Existing Personnel With Mainframes—CICS Experience?

CICS/6000 has a programming interface similar to mainframe CICS. The learning curve for mainframe CICS programmers to use CICS/6000 would be minimal; for these same personnel to program using TUXEDO, Encina, or TOP END, the learning curve would be substantial. On the other hand, because CICS/6000's administrative facilities are not similar to mainframe CICS, administrative personnel will face a steep learning curve: they will need to learn UNIX, DCE, and Encina (the layers on which CICS/6000 is built). (NOTE: VIS/TP and UniKix are also implementations of CICS in the UNIX environment, but they ere not included in this evaluation.)

Possible Product Options

Tuxedo; CICS/6000; Encina; MS Transaction Server; Sybase Jaguar; TOP END; openUTM; TransIT Open/OLTP

Below are commonly used transaction monitors:

BEA TUXEDO—provides a robust middleware engine for developing and deploying business-critical client/server applications. BEA TUXEDO handles not only distributed transaction processing, but also application and the full complement of services necessary to build and run enterprise-wide applications. It enables developers to create applications that span multiple hardware platforms, databases and operating systems.

IBMs CICS/6000—an application server that provides industrial-strength, online transaction processing and transaction management for mission-critical applications on both IBM and non-IBM platforms. CICS manages and coordinates all the different resources needed by applications, such as RDBMSs, files and message queues to ensure completeness and integrity of data.

Transarcs Encina—implements the fundamental services for executing distributed transactions and managing recoverable data, and various Encina extended services, which expand upon the functionality of the toolkit to provide a comprehensive environment for developing and deploying distributed transaction processing.

Microsofts Transaction Server (Viper)—a component-based transaction processing system for developing, deploying, and managing high performance, and scalable enterprise, Internet, and intranet server applications. Transaction Server defines an application programming model for developing distributed, component-based applications. It also provides a run-time infrastructure for deploying and managing these applications.

Brief Product Considerations

Encina—The Encina DTP (OLTP) was built on top of OSF's DCE. This is both its greatest asset and curse. DCE offers a very complete set of functions including security services, RPC's, a directory service (like a yellow pages for clients to find services) and a standard time service, and it is truly cross-platform and is endorsed by most vendors. The problem is that it is a resource hog, and is fairly slow. DCE is also somewhat immature in that there are not many tools to help you administer or program applications (although many are on the way). Encina adds primarily a transactional element and some load balancing services to RPC's. It also provides an easier interface to work with (although it is still an administrative nightmare).

The good news is that the tools are getting better all of the time. Also, Encina is very scalable and services can be on any machine in the network. Finally, Encina's load balancing is quite good, much better then native DCE or Tuxedo.

Tuxedo

Functionality

Can handle a large number of concurrent client applications

Can handle a large volume of through-put (ex. 1000+TPS)

Scaleable (handle many clients or a few without code rewrite)

Supports Transactions, including XA transactions

Has its own transaction resource manager

Guaranteed message delivery using a stable storage queue (/Q)

Future service delivery using/Q (usually for batch processing)

Can prioritize messages- most important get processed sooner.

Supports many platforms (all UNIX, NT, all common client platforms)

Tuxedo supports C, C++, and Cobol development

Can be used for basic c/s messaging

Supports conversational messaging between a client and a specific server

Peer-to-peer, client-to-client messaging is supported

Unsolicited messaging is supported for client processes

Asynchronous service calls can be made by client and server processes

Synchronous service calls can be made by client and server processes

Synchronous calls that receive no return message are supported

Very good security- must connect to access services

Security can be integrated w/Kerberos

Has many different buffer types: view to pass C structs, FML to pass anything, carrays to pass binary (sound, video), strings to pass strings

FML allows dynamic messages to be sent/received

Automatic error logging for Tuxedo components (ULOG, tagent log)

Application code can write to the ULOG with a Tuxedo API (error logging provided)

Automatic process monitor for process that die or machines that get partitioned

Service location independency (distribution/directory services)

Platform independency-handles data conversion

Built in data compression (if desired)

Built in performance measurement feature for services

Built in admin functions to monitor Tuxedo system online (tmadmin)

A server can be called based on data in the message (Data Dependent Routing)

Customizable server start-up and shutdown functions are automatically called.

/Domains allow independent Tuxedo regions to share services

Extensions to execute IMS and CICS transactions from UNIX (Open Transport)

Subscribe and Broadcast supported

APIs to get admin and system monitoring data for custom operation tools

JOLT (java to access Tuxedo servers)

Other Reasons to Use Tuxedo

Tuxedo is the market leader OLTP

Tuxedo is a proven product used in mission critical systems govt. and financial)

Tuxedo can be used to develop highly-available systems (24×7)

Has been implemented with PowerBuilder, VisualBasic, Motif clients, and unix batch systems.

Cons of Using Tuxedo

Tuxedo for basic c/s messaging is overkill.

Expensive to purchase

Can be complicated to develop with and administer

System performance tuning requires an experienced Tuxedo administrator

Uses IPC resources and therefore should not be on same machine w/other IPC products

Must be understood thoroughly before design starts. If used incorrectly, can be very costly.

Single threaded servers requires an upfront packaging design.

Difficult to debug servers

Does not work well with Pure Software products: Purify, Quantify

Servers must be programmed to support client context data management

Difficult to do asynch messaging in 3rd party Windows 3.x client tools (ex. PowerBuilder)

Resource Management **2604**

A Resource Manager provides for concurrency control and integrity for a singular data resource (e.g., a data base or a file system). Integrity is guaranteed by ensuring that an update is completed correctly and entirely or not at all. Resource Management Services use locking, commit, and rollback services, and are integrated with Transaction Management Services.

Transaction Manag ement **2606**

Transaction Management Services coordinate transactions across one or more resource managers either on a single machine or multiple machines within the network. Transaction Management Services ensure that all resources for a transaction are updated, or in the case of an update failure on any one resource, all updates are rolled back.

This services that allow multiple applications to share data with integrity. The transaction management services help implement the notion of a transaction—a set of computations producing changes to recoverable data which demonstrate the ACID properties:

Atomicity—all changes are made completely (committed) or not at all (roll-back).

Consistency—the effects of a transaction preserve invariant properties.

Isolation—intermediate data values are not visible to other transactions.

Durability—the effect of a completed transaction are persistent.

Two-Phase Commit is a feature found in distributed database management systems and online transaction processing (OLTP) monitors to ensure information integrity across distributed databases. With this feature, a transaction is only commited if two databases have the necessary information. If a problem arises on a network connection or a computer, the software will roll the transaction back so it will not be entered in either place. A restart mechanism may then retriy to complete the transaction.

Possible Product Options

Tuxedo; Encina; TOP END; CICS/6000; openUTM; TransIT Open/OLTP

Transaction Partitioning **2608**

Transaction Partitioning Services provide support for mapping a single logical transaction in an application into the required multiple physical transactions. For example, in a package or legacy rich environment, the single logical transaction of changing a customer address may require the partitioning and coordination of several physical transactions to multiple application systems or databases. Transaction Partitioning Services provide the application with a simple single transaction view.

Implementation Considerations

Must the System Support Logical Transactions That Occur Across Heterogenous Application Servers and Databases?

EXAMPLE

In a given application, a single business process of updating a customer record requires an update to a table in

a UNIX based relational database and then an update to a table in a MVS DB2database. Although there are two physical transactions occurring, this entire business process is represented as a single logical transaction. Transaction Partitioning services allow the application to ensure that the individual transactions occurr across the different UNIX and MVS systems and that the single logical transaction is completed and successful when the individual physical transactions are completed and successful.

ENVIRONMENT **1016,1018**

FIG. **27** illustrates various components of the Environmental Services of the Netcentric Architecture Framework. Environment Services provide miscellaneous application and system level services that do not deal directly with managing the user-interface, communicating to other programs, or accessing data.

Runtime Services **2702**

Runtime services convert non-compiled computer languages into machine code during the execution of a program.

Language Interpreter **2704**

Language Interpreter Services decompose a 4th generation and/or a scripting languages into machine code (executable code) at runtime.

Possible Product Options

VBRUN300.DLL

VBRUN300.DLL—runtime Dynamic Link Library that supports programs written in Visual Basic.

Virtual Machine **2706**

Typically, a Virtual Machine is implemented in software on top of an operating system, and is used to run applications. The Virtual Machine provides a layer of abstraction between the applications and the underlying operating system and is often used to support operating system independence.

Possible Product Options

Java virtual machine; Smalltalk virtual machine

Virtual machines such as the Java virtual machine or the Smalltalk virtual machine implement their own versions of operating system services in order to provide the application with complete platform independence.

Java virtual machine—software implementation of a "CPU" designed to run compiled Java byte code. This includes stand-alone Java applications as well as "applets" that are downloaded and run in Web browsers.

Smalltalk virtual machine—runtime engine that interprets application code during execution and supports platform independence.

System Services **2708**

Services which applications can use to perform system-level functions. These services include: System Security Services, Profile Management Services, Task and Memory Management Services, and Environment Verification Services.

System Security **2710**

System Security Services allow applications to interact with the operating system's native security mechanism. The basic services include the ability to login, logoff, authenticate to the operating system, and enforce access control to system resources and executables.

Profile Management **2712**

Profile Management Services are used to access and update local or remote system, user, or application profiles. User profiles, for example, can be used to store a variety of information such as the user's language and color preferences to basic job function information which may be used by Integrated Performance Support or Workflow Services.

Implementation Considerations

Is There a Need for the Application to Have its own Profile File?

All MS Windows based application maintain their own profile file (XXXXXXXX.INI) that is used during application startup, execution, and shutdown. This is a flat text file that contains information that can be used by the application during various phases of execution. For example, if an application needs to connect to a database engine/server, it needs to know, during startup, various information like—database name, the server name, login ID, etc. Instead of hard coding all these information in the application executable, this information can be stored in the profile file for flexibility. In the future, if the database server name should change, this change only needs to be entered in the applications profile file.

In some cases, it has been seen that this profile information has been hard coded in that applications executable itself. This will work, but, it makes the application more rigid with no room for any flexibility.

Environment Verification **2714**

Environment Verification Services ensure functionality by monitoring, identifying and validating environment integrity prior and during program execution. (e.g., free disk space, monitor resolution, correct version). These services are invoked when an application begins processing or when a component is called. Applications can use these services to verify that the correct versions of required Execution Architecture components and other application components are available.

Implementation Considerations

In client/server applications, it may be necessary to implement Environment Verification Services to ensure that the client and server applications are of a compatible release level.

ActiveX framework provides services for automatic installation and upgrade of ActiveX controls. When using IE, i.e., Microsoft's Web browser, because of its integration with Windows OS, ActiveX controls can be automatically installed and automatically upgraded on the users machine without the developer adding any additional code.

Task and Memory Management **2716**

Task & Memory Management Services allow applications and/or other events to control individual computer tasks or processes, and manage memory. They provide services for scheduling, starting, stopping, and restarting both client and server tasks (e.g., software agents).

Implementation Considerations

Memory management, the allocating and freeing of system resources, is one of the more error prone development activities when using **3**GL development tools. Creating architecture services for memory handling functions can reduce these hard to debug errors.

Java removes, in theory, the problem of memory management, by providing a garbage collector; although, its implementation is not very efficient in current implementations of Java. Future releases of the Java VM promise a background-running garbage collector with significantly increased performance.

Application Services **2718**

Application Services are miscellaneous services which applications can use for common functions. These common functions can apply to one application or can be used across applications. They include: Application Security Services, Error Handling/Logging Services, State Management Services, Help Services, and Other Common Services.

Application Security **2720**

Besides system level security such as logging into the network, there are additional security services associated with specific applications. These include:

- User Access Services—set of common functions that limit application access to specific users within a company or external customers.
- Data Access Services—set of common functions that limit access to specific data within an application to specific users or user types (e.g., secretary, manager).
- Function Access Services—set of common functions that limit access to specific functions within an application to specific users or user types (e.g., secretary, manager).

Implementation Considerations

In the Netcentric environment, application security becomes a more critical component primarily because there are more types of users (e.g., employees, customers) and additional types of transactions (e.g., e-commerce, help-desks). In traditional client/server environments most users are employees of the company. In Netcentric environments there are typically also external users (e.g., vendors, registered users) and the general public. Usually, different types of users have different application security requirements limiting what data they can see and what functions they can execute. Also, new types of transactions such as verifying credit when doing e-commerce transactions also require additional application security services.

Error Handling/Logging **2722**

Error Handling Services support the handling of fatal and non-fatal hardware and software errors for an application. An error handling architecture takes care of presenting the user with an understandable explanation of what has happened and coordinating with other services to ensure that transactions and data are restored to a consistent state.

Logging Services support the logging of informational, error, and warning messages. Logging Services record application and user activities in enough detail to satisfy any audit trail requirements or to assist the systems support team in recreating the sequence of events that led to an error.

Implementation Considerations

Error Handling

Primarily there are three types of errors: system, architecture and application.

- System errors occur when the application is being executed and some kind of serious system-level incompatibility is encountered, such as memory/resource depletion, database access problems, network problems or printer related problems, because of which the application cannot proceed with its normal execution.
- Architecture errors are those which occur during the normal execution of the application and are generated in architecture functions that are built by a project architecture team to isolate the developers from complex coding, to streamline the development effort by re-using common services, etc. These architecture functions perform services such as database calls, state management, etc.
- Application errors are also those which occur during the normal execution of the application and are generally related to business logic errors such as invalid date, invalid price, etc.

Typically an application is written using a combination of various programming languages (e.g., Visual Basic and C). Therefore, a common error handling routine should be written in a language that can be called from any other language used in the application.

Logging

Logging must be done, however to mitigate problems, centralize logs and create a standard, usable log format. 3rd party logs should be mapped into the central format before any analysis is attempted.

In a Netcentric environment, errors are rarely logged on the client machine (one exception may be for an intranet type application).

Logging can add much stress to a Web server and logs can grow very large, very quickly, so do not plan to log all errors—capture only those which are deemed necessary for processing exceptions.

State Management **2724**

State Management Services enable information to be passed or shared among windows and/or Web pages and/or across programs. So lets say several fields in an application need to be passed from one window to another. In pseudo-conversational mainframe 3270 style applications passing data from one screen to another screen was done using Context Management Services that provided the ability to store information on a host computer (in this paper the term Context Management refers to storing state information on the server, not the client). Client/server architectures simplified or eliminated the need for Context Management (storing state information on the server), and created a need to store state information on the client. Typically, in traditional client/server systems this type of state management (i.e., data sharing) is done on the client machine using hidden fields, global variables, messages, files or local databases.

The popularity of the Internets HTTP protocol has revived the potential need for implementing some form of Context Management Services (storing state information on the server). The HTTP protocol is a stateless protocol. Every connection is negotiated from scratch, not just at the page level but for every element on the page. The server does not maintain a session connection with the client nor save any information between client exchanges (i.e., web page submits or requests). Each HTTP exchange is a completely independent event. Therefore, information entered into one HTML form must be saved by the associated server application somewhere where it can be accessed by subsequent programs in a conversation.

Advances in Netcentric technologies now offer additional options for implementing state management on!both the client and server machines.

Possible Product Options

NetDynamics Inc. NetDynamics

NetDynamics Inc. NetDynamics

NetDynamics provides built-in, developer-definable session and state management. The Persistence Engine (PE), part of the NetDynamics application server, stores all relevant information about a user. Everything from the WebID to the exact table row the user is currently viewing can be maintained in the PE. NetDynamics maintains state information on both the server and on the client page. Application state information is maintained by the application server, and local state information is maintained on the page. NetDynamics provides manipulatable state objects for both server and page state information.

Codes Table Service **2726**

Codes Table Services enable applications to utilize externally stored parameters and validation rules. For example, an application may be designed to retrieve the tax rate for the State of Illinois. When the user enters "Illinois" on the screen, the application first validates the user's entry by

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 176 of 281

checking for its existence on the "State Tax Table", and then retrieves the tax rate for Illinois. Note that codes tables provide an additional degree of flexibility. If the tax rates changes, the data simply needs to be updated; no application logic needs to be modified.

Implementation Considerations

Is There a Need for the Codes Table Functionality?

Most applications need code/decode facility. For example, an application may need to store codes like—error severity codes, etc., stored in a table (may be a cached table) instead of in the executable itself. In some cases, where there is a small amount of information that needs to be stored in the codes table, the profile file (mentioned above) can be used instead of the codes table. But in cases where the codes table needs to be used quite extensively, then storing the code/decode information in the profile file will slow down the performance of the application because of the overhead of accessing flat files.

What Basic Services an Architecture Should Provide in Terms of Managing/Using Codes/decodes Functionality?

In cases where the application requires extensive use of codes table, the architectures Code/Decode component should provide the application developers with a set of API that can be used to use code/decode tables. This component should also provide the option of caching all or parts of the codes table in the application machines memory for easier and faster access.

Where Should Code/Decode Information be Stored and Maintained?

Code/decode information can be stored at any layer of an n-tier architecture—client, application server, or database. The decision will need to be based upon codes table size and number, information update frequency, and write-access to the client machine or device.

Active Help **2728**

Active Help Services enable an application to provide assistance to a user for a specific task or set of tasks. Context-sensitive help is most commonly used in applications today, however this can imply more "active" support that just the F1 key. Typically, today's systems must be architected to include Help that is aware of both the user's environment, process and context, and in this sense can be called "active". Active Help services may include components like Wizards for walking a user through a new process, stored or real-time multi-media support, on-demand Computer Based Training, etc.

Other Common Services **2726**

Catchall category for additional reusable routines useful across a set of applications (e.g., Date Routines, Time Zone Conversions, Field Validation Routines).

Implementation Considerations

Does the Client Operate in Different Date/time Zone?

In most large scale distributed applications, the client and the server applications (or machines) are scattered over different time zones. This forces the client applications and the server hosts to deal with date and time zone conversions (like -CST to PST, etc.) in order to use or display their local time accurately. Most of the architectures provide a base set of APIs that can be used by the applications to convert the data/time as needed.

Does the System Requires Customized Date/time Format for Display Purposes?

Many systems, for certain business reasons, need customized date and time formats for display and storage purposes. In order to do that, the architecture should provide a set of APIs that will allow the system to convert data and time from one format to the other.

Does the System Deal with High Database Accesses?

As mentioned in the Codes Table Component, sometimes it is necessary to cache the data in the memory for faster access and less database hits. This a feature that some architectures provide as a set of memory management APIs to create the cache area in the client platforms memory for the data to reside.

Application Integration Interface **2734**

An Application Integration Interface provides a method or gateway for passing context and control of information to an external application. The Application Integration Interface specifies how information will be passed and defines the interface by which other applications can expect to receive information. External applications in this context could include anything from Integration Performance Support systems to ERP systems like SAP or Peoplesoft to external custom applications that have been previously developed by the client.

Implementation Considerations

Where possible, Application Integration Interfaces should make use of the Component Model defined by the project to broker information (i.e. OLE/COM interfaces) as opposed to custom building data sharing modules.

Component Framework **2736**

Component Framework Services provide an infrastructure for building components so that they can communicate within an application and across applications, on the same machine or on multiple machines across a network, to work together. COM/DCOM and CORBA described in Communication Services are the two leading component industry standards. These standards define how components should be built and how they should communicate.

Object Request Broker (ORB) services, based on COM/DCOM and CORBA, focus on how components communicate. Component Framework Services, also based on CORBA and COM/DCOM, focus on how components should be built. The currently 2 dominant Component Frameworks include:

1. ActiveX/OLE—ActiveX and Object Linking and Embedding (OLE) are implementations of COM/DCOM. ActiveX is a collection of facilities forming a framework for components to work together and interact. ActiveX divides the world into two kinds of components: controls and containers. Controls are relatively independent components that present well defined interfaces or methods that containers and other components can call. Containers implement the part of the ActiveX protocol that allows for them to host and interact with components—forming a kind of back plane for controls to be plugged into. ActiveX is a scaled-down version of OLE for the Internet. OLE provides a framework to build applications from component modules and defines the way in which applications interact using data transfer, drag-and-drop and scripting. OLE is a set of common services that allow components to collaborate intelligently.

In creating ActiveX from OLE 2.0, Microsoft enhanced the framework to address some of the special needs of Web style computing. Microsofts Web browser, Internet Explorer, is an ActiveX container. Therefore, any ActiveX control can be downloaded to, and plugged into the browser. This allows for executable components to be interleaved with HTML content and downloaded as needed by the Web browser.

2. JavaBeans—is Sun Microsystems proposed framework for building Java components and containers. The intent is to develop an API standard that will allow components developed in Java (or beans), to be embed-

ded in competing container frameworks including ActiveX or OpenDoc. The JavaBeans API will make it easier to create reusable components in the Java language.

Other component frameworks include:

OpenDoc—CI Labs was formed in 1993 and created the OpenDoc architecture to provide a cross-platform alternative component framework—independent of Microsofts OLE. The OpenDoc architecture is constructed from various technologies supplied by its founding members—IBM, Apple and Word Perfect. The technologies include: Bento (Apples object storage model), Open. Scripting Architecture (OSA—Apples scripting architecture) and SOM/DSOM (IBMs System Object Model/Distributed SOM). IBMs SOM architecture provides analogous services to that of Microsofts DCOM architecture.

OpenDoc provides an open compound document infrastructure based on CORBA. It uses CORBA as its object model for inter-component communications. OpenDoc architecture provides services analogous to those provided by OLE and OpenDoc components can also inter-operate with OLE components. The OpenDoc equivalent of an object is termed a part. Each type of part has its own editor and the OpenDoc architecture has responsibility for handling the communications between the distinct parts.

Supporters claim OpenDoc provides a simpler, more technically elegant solution for creating and manipulating components than does OLE. The drawback is that OpenDoc is not yet commercially proven, like OLE. Ironically, one of the more popular uses of OpenDoc tools is for creating and implementing OLE clients and servers. Because OpenDoc provides a more manageable set of APIs than OLE, it may be that OpenDoc gains initial acceptance as an enabler of OLE applications before becoming recognized as a complete component software solution itself.

ONE—Open Network Environment (ONE) is an object-oriented software framework from Netscape Communications for use with Internet clients and servers, which enables the integrating of Web clients and servers with other enterprise resources and data. By supporting CORBA, ONE-enabled systems will be able to link with object software from a wide array of vendors, including IBM, Sun Microsystems, Digital Equipment, and Hewlett-Packard. Netscape is positioning ONE as an alternative to Microsoft's Distributed Common Object Model (DCOM). ONE also complies with Sun Microsystems Java technology.

Implementation Considerations

An architecture that utilizes components brings many of the benefits of object orientation to applications. Component-based or document-centric applications are composed of intelligent components, each of which contains logic, possibly data and a set of well defined interfaces or APIs to the services they provide (e.g., a customer component or an Excel chart component). The similarities to object oriented are more than just coincidental. Component software is viewed by many as a more viable object approach focusing on larger grain of modularity and reuse.

Two important issues driving the decision around what should be a component are software re-use and software packaging. Software re-use will primarily stem from defining components at a level at which they can be re-used within the same application and across many applications. Although re-usable components can be at any level, more often they will probably be at an object level where they are more granular. Software packaging will be driven by defining components at a level at which they can be distributed efficiently to all users when business logic changes occur. If the application is large, perhaps it is better to package the application by breaking it up into process components such as customer maintenance, sales order maintenance, etc. So when a change to one of the processes occurs, only the component which contains that process needs to be distributed to client machines, rather than the whole application. For example, a developer can create an ActiveX control that will encapsulate the Employee Maintenance Process, which includes adding new employees, updating and deleting existing employees. This ActiveX control can be a part of an overall human resource intranet application. When the functionality within the Employee Maintenance Process changes, the next time the user accesses the human resource application from the Web browser, ActiveX technology will automatically download the latest version of the ActiveX control containing the most recent update of the Employee Maintenance Process to the client machine, if the client machine does not have the latest version.

Component Architectures Typically Employ of a Three-tier Component Architecture Utilizing the Following Three Types of Components:

User Interface, Process, and Domain. While these three component types may go by different names on different projects, they all follow the same basic pattern and are briefly explained below:

User Interface components typically contain nothing more than the logic required to manipulate input and output to the user. This can include input validation requiring no additional server data, as well as simple calculations associated with field display. In addition, logic associated with dynamically changing the display (e.g., a checkbox entry causes a field to become disabled) is placed here.

Process components typically contain the logic associated with business transactions performed on data. This is often the point where transaction commit/rollback occurs. These components are typically invoked by the User Interface components.

Domain components typically contain the logic associated with accessing and maintaining business entities, i.e., data. These components are usually invoked by Process components when requiring access to or manipulation of data. However, in addition to data access, these components may often be used to perform manipulations involving the processing of data within the domain of that component. For example, a Customer Domain component might be requested to determine if it's credit limit had been exceeded when provided with a new invoice amount.

Build vs. Buy

There is an explosion of components available in the market place and the ease of accessing and down loading components from the Internet; the decision to buy or build a component is as real as ever. In general clients expect more justification of a build decision v. a buy decision. Feel out the client and the expectations and requirements they may have.

Components are a viable option and should be researched, even includingeven simple UI controls available on the Internet. Look at market trends to determine which applications/components can meet the bulk of the system needs.

Operating System **2738**

Operating System Services are the underlying services such as multi-tasking, paging, memory allocation, etc.,

typically provided by today's modem operating systems. Where necessary, an additional layer or APIs may be provided to gain either operating system independence or a higher level of abstraction for application programmers.

Possible Product Options

Microsoft Windows; Windows 95; Windows NT; Macintosh OS; OS/2 ; Unix and Java OS

BASE SERVICES **1020**

Component Description

FIG. **28** illustrates the Base Services of the Netcentric Architecture Framework. Base Services provide server-based support for delivering applications to a wide variety of users over the Internet, intranet, and extranet. The information about these services in the Netcentric framework may be limited based on the least common denominator. For more detailed information about these components refer also to the following frameworks in SAF and/or DAF.

Batch Delivery Vehicle

Collaboration Framework for Structured Information (Workflow)

Web Services **(2820)**

Web Server Services enable organizations to manage and publish information and deploy Netcentric applications over the Internet and intranet environments. These services support the following:

Managing documents in most formats such as HTML, Microsoft Word, etc.

Handling of client requests for HTML pages. A Web browser initiates an HTTP request to the Web server either specifying the HTML document to send back to the browser or the server program (e.g., CGI, ASP) to execute. If the server program is specified, the Web server executes the program which generally returns a formatted HTML page to the Web Server. The Web server then passes this HTML page just as it would any standard HTML document back to the Web browser.

Processing scripts such as Common Gateway Interface (CGI), Active Server Pages (ASP). Server side scripting enables programs or commands to be executed on the server machine providing access to resources stored both inside and outside of the Web server environment. For example, server side scripts can be used to process requests for additional information, such as data from an RDBMS.

Caching Web pages. The first time a user requests a Web page, the Web server retrieves that page from the network and stores it temporarily in a cache (memory on the Web server). When another page or the same page is requested, the Web server first checks to see if the page is available in the cache. If the page is available, then the Web server retrieves it from the cache, otherwise it retrieves it from the network. Clearly, the Web server can retrieve the page from the cache more quickly than retrieving the page again from its location out on the network. The Web server typically provides an option to verify whether the page has been updated since the time it was placed in the cache, and if it has to get the latest update.

Possible Product Options

Netscape Enterprise Web Server; Microsoft Internet Information Server (IIS); Oracle WebServer

The following are relevant products for providing or implementing HTTP Web Server Services:

Netscape Enterprise Web Server

An enterprise-strength Web server that enables organizations to manage and publish their information and deploy Netcentric applications. Netscape Enterprise Web Server is built on open Internet standards that enable information and applications to scale easily. Supports S-HTTP, Java, and SNMP.

Microsoft Internet Information Server (IIS)

A free add-on product for NT Server that implements basic HTTP services. Future versions of NT Server (4.0 and beyond) will have HTTP features built directly into the operating system.

Oracle WebServer

A multi-threaded HTTP server that provides integrated features for translating and dispatching client HTTP requests directly to the Oracle**7** Server using PL/SQL.

Push Pull Services **(2840)**

Push/Pull Services allow for interest in a particular piece of information to be registered and then changes or new information to be communicated to the subscriber list. Traditional Internet users "surf" the Web by actively moving from one Web page to another, manually searching for content they want and "pulling" it back to the desktop via a graphical browser. But in the push model, on which subscription servers are based on, content providers can broadcast their information directly to individual users' desktops. The technology uses the Internet's strengths as a two-way conduit by allowing people to specify the type of content they want to receive. Content providers then seek to package the requested information for automatic distribution to the user's PC.

Depending upon requirements, synchronous or asynchronous push/pull services may be required. Synchronous push/pull services provide a mechanism for applications to be notified in real time if a subscribed item changes (e.g., a stock ticker). Asynchronous push/pull services do not require that a session-like connection be present between the subscriber and the information. Internet ListServers are a simple example. Subscribers use e-mail to register an interest in a topic and are notified via e-mail when changes occur or relevant information is available. Asynchronous push/pull services can be useful for pro-actively updating customers on changes in order status or delivering information on new products or services they have expressed an interest in.

PointCast; Marimba; IBM/Lotus; Microsoft; Netscape; America Online; BackWeb; Wayfarer

Castanet from Marimba—distributes and maintains software applications and content within an organization or across the Internet, ensuring subscribers always have the most up-to-date information automatically.

PointCast—news network that appears instantly on the subscribers computer screen.

Batch Services (B**2060**)

Batch processing is used to perform large scale repetitive processing where no user involvement is required as well as reporting. Areas for design attention include scheduling, recovery/restart, use of job streams and high availability (e.g. 24 hour running). In addition close attention must be paid to performance as batch systems usually must be processed within strict batch windows.

The design of batch architectures is often complicated considerably by the fact that batch jobs must be able to run concurrently with on-line systems. The general globalization of companies requires that he on-line systems must be available on a close to 24×7 hours basis, eliminating the traditional batch windows. Concurrent batch and on-line processing poses serious challenges to data integrity, throughput and performance.

Batch application programs can include business processing such payroll, billing, etc. and can also include report

generation. This is an often overlooked area in client/server architectures. Traditional client/server solutions and Netcentric solutions often require batch processing, but unlike the mainframe, the typical platforms and development environments used often do not have built-in batch or reporting architecture facilities.

Batch processing should be used in preference to on-line modules when:

- The same process, or set of processes, must be applied to many data entities in a repetitive and predictable fashion.
- There is either no manual element to the process or the manual element can be completely separated from a batch element.

The volume of information to be presented to a user is too great to be processed on-line or it can be better printed in batch.

Related Patterns

For more detailed information about component based batch design patterns, refer also to the Batch patterns in the Patterns section:

- Base Services Patterns Overview
- Abstraction Factory
- Batch Job
- BUW—Batch Unit of Work
- Processing Pipeline

Report Services **(2880)**

Report Services are facilities for simplifying the construction and delivery of reports or generated correspondence. These services help to define reports and to electronically route reports to allow for online review, printing, and/or archiving. Report Services also support the merging of application data with pre-defined templates to create letters or other printed correspondence. Report Services include:

- Driver Services. These services provide the control structure and framework for the reporting system.
- Report Definition Services. These services receive and identify the report request, perform required validation routines, and format the outputted report(s). After the request is validated, the report build function is initiated.
- Report Build Services. These services are responsible for collecting, processing, formatting, and writing report information (for example, data, graphics, text).
- Report Distribution Services. These services are responsible for printing, or otherwise distributing, the reports to users.

FUNCTIONS AND FEATURES OF A REPORT ARCHITECTURE

The report architecture within Environment Services supports the generation and delivery of reports. Applications request report services by sending a message to the reporting framework.

The following types of reports are supported by the reporting application framework:

- Scheduled: Scheduled reports are generated based upon a time and/or date requirement. These reports typically contain statistical information and are generated periodically (invoices and bills, for example).
- On-demand: Some reports will be requested by users with specific parameters. The scheduling of these reports, the formatting, and/or the data requirements are not known before the request is made, so these factors must be handled at re quest time.
- Event-driven: This report type includes reports whose generation is triggered based on a business or system event. An example here would be a printed trade slip.

REPORTING APPLICATION FRAMEWORK

FIG. **29** shows the major components of the reporting application framework:

Report Initiation **(2900)**

The report initiation function is the interface for reporting applications into the report architecture. The client initiates a report request to the report architecture by sending a message to the report initiation function. The responsibility of report initiation is to receive, identify, and validate the request and then trigger the report build process. The main components of reporting initiation are the following.

- Receive, identify, and validate a report request. The identification function determines general information about the request, such as report type, requester, quantity to be printed, and requested time. Based on the report type, a table of reports is examined in order to gather additional report-specific information and perform required validation routines for the report request. After the report identification and validation functions have been successfully completed, the reporting process can continue. If any errors are identified, the report initiation function will return an error message to the requester application.

Initiate report execution. The initiate report execution function processes the report profile and specific distribution requirements and determines the report to be created. It then passes control to the report execution process.

Report Execution **(2902)**

Report execution is the core of the reporting application framework. The main components of report execution include:

- Format the report. This function is responsible for formatting the layout of the outputted report, including standard headers, column headings, row headings, and other static report information.
- Collect the information. This function is responsible for collecting the information (for example, data, text, image, graphics) that is required for the report. This function would utilize the Information Access Services component of the client/server architecture.
- Format the information. This function is responsible for formatting the collected information into the appropriate display format based upon the report type and the report distribution requirements.
- Output the report. This function initiates the report distribution function in order to distribute the created report to the specified devices (printers, disks, and so forth) and individuals.

The process of collecting, processing, formatting, and outputting report data can be accomplished in several different ways. For example, one method is to create a program in C for each report format. Here, many aspects of report printing—such as page size, headings, footings, and printer control values—would have to be programmed in function calls to facilitate the report programming process. Information access to files or the database would be through Information Access Services.

Another option is to use a third-party report tool, such as the SQR (Structured Query Report Writer) from SQL Solutions. SQR is a robust report generator designed to be used with SQL-based relational databases. SQR insulates the developer from programming in a third generation language

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 180 of 281

by providing a higher-level programming language. SQL queries (Information Access) are placed directly into the SQR program.

Report Distribution (**2904**)

The final requirement of the reporting application framework is the report distribution function. Once the report has been generated, it must be distributed to the specified targets (devices and/or users). The report distribution function will locate completed report files and route them to the appropriate devices within the client/server network.

Typically, a report distribution database is used to specify the destinations for each report supported by the, report architecture. The report distribution database specifies where, when, how, and to whom to distribute the produced report. Specific destinations can include: printer(s), user(s), user groups, archives (permanent storage), and/or specific display devices such as workstations and terminals.

Several additional options exist for distributing reports including timed reporting, multiple copy distribution, and report archiving. Also, a user interface function can be built to open and browse report files.

CUSTOM REPORTING APPROACHES

If a commercially-available reporting product can not meet your report requirements, you may have to consider a custom approach. FIG. **30** illustrates an example of how a custom report architecture relates to a workstation platform technology architecture.

This custom report process is responsible for processing all messages requesting generation, manipulation, or distribution of reports. The following services are provided in an environment including a pair of workstations **3000** and a server **3002**:

Report generation

Report deletion

Report printing

Report status maintenance

Report generation is supported by an additional report writer process that contains all application-defined report writer modules. These modules contain the logic to produce each of the report types that may be requested. The report process receives generation requests and ensures that they are forwarded to the report writer process at the current or specified time. All report requests are processed in an asynchronous manner (for example, service requesters do not wait for completion of report processing).

FIG. **31** describes the relationships between the major components of the report process **3100** and the report writer process **3102**.

Design Approach

For the report process in a client/server system, a set of APIs is provided for use within application programs and within the application report writer modules. Each API requests a specific report service (generation, printing, or deletion) which is performed by a report manager module.

The report process maintains an internal database table, a report status table, containing information about each report that has been requested for generation, including:

Requester ID

Report name

Date/time requested

Status (requested, in process, complete, or error)

Report-specific parameters.

The requester ID, report name, and date/time are used to uniquely identify the report. These values are passed to APIs which request report status, print or delete a previously generated report.

All application-defined report writer modules invoke an API to update the report status table with a status of "completed" after a report has been produced or with "error" if the report cannot be generated. An API is also provided to print the report after the generation if specified in the original request.

Processed report records are removed from the table only after the output reports have been archived. Implementation and frequency of this table cleanup is to be determined in systems management design.

Report Process Flows

Report processing is message-driven. Each defined API sends a unique message to the report process. The report process reads the messages from a queue and invokes the appropriate modules to handle each request. Subsequent process flows differ based upon the requested service. In the case of a report generation request, the process flow proceeds as follows:

A record is added to the report status table.

A message is sent to the report writer process for immediate generation or to the event manager for generation at a specified time (report scheduling).

The appropriate application report writer module generates the report, prints it if specified in the original API request, and updates the status in the report status table.

A request to print a report proceeds as follows:

The report status is retrieved from the report status table.

The output file is located on disk and sent to the specified or default printer or the request is sent to the event manager for report scheduling.

Report deletion proceeds as follows:

The report record is removed from the report status table.

The report file is removed from disk.

Status information requests are performed directly from the API using Information Access Services APIs. No interaction with the report process is necessary, which results in improved performance.

Modules

FIG. **32** shows the module hierarchy for the custom report process. The Figure shows the relationships between modules, not their associated processing flows. It should be used to identify the calling module and the called modules for the process. FIG. **32** illustrates the Architecture Manager library **3200** which supports the report process.

The functions designed to support this process are:

Generate Report

Get Report Status

Control Reports

Request Report (b**2402**)

Delete Report (b**2406**)

Print Report (b**2404**)

Generate Report. This module is called to request report generation and printing (optional). Input data blocks specify the following:

Report name

Report parameters

Report generation time (default is immediately)

Printer name.

The report name must be one of the defined application report types. Valid report parameters vary depending on the report type. Reports may be requested for generation immediately or at a designated future time. All reports are written

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 181 of 281

to a reserved area on disk; however, specification of a printer causes the output to be printed as well as stored on the file system.

Get Report Status. The Get Report Status function retrieves status information about all reports that have been previously requested for generation by the calling process. Returned is a list containing the requested data as well as the number of reports found.

Control Reports. The Control Reports function is responsible for performing various operations on reports. The following services are provided:

Delete a report request and any associated output

Print a previously generated report.

Update report status.

In all cases, the report name is passed through an input data block. For the print service, a printer name is passed. For status update, the new status code is passed.

Request Report. The Request Report function is responsible for processing report request messages written to the report process queue. It creates a new entry in the report status table with a status of "requested" and initiates the report writer process for immediate generation or sends a message to the event manager for future report generation.

Delete Report. The Delete Report function is responsible for removing a report from the Report Status list and deleting the generated output file (if any).

Print Report. The Print Report function sends a generated report output file to a specified or default printer. The report name and requesting process ID is passed to identify the report.

EVALUATION CRITERIA

There are two primary approaches to implementing a reporting architecture: custom and package. Evaluating custom and package solutions involves both functional and technical criteria. The following is a discussion of various functional and technical criteria that should be considered during the planning for a report architecture. Note that not all of the criteria may be required by any particular organization.

Functional Criteria.

1. Report Repository: The report architecture should work with, and support maintenance of, a report repository on the platforms within the client/server architecture. The report repository contains the detailed definitions of the reports.

2. Workgroup Report Support: The report architecture should work with and support distribution of reports generated on the workgroup server.

3. On-Demand Reports: The report architecture must support distribution of reports requested by users on demand. Typically, these reports will not have a set schedule or frequency for distribution. The report architecture must support distribution of these reports without the requirement of manual or user intervention (subsequent to initial set up and conversion).

4. Scheduled Reports: The report architecture must support distribution of regularly scheduled reports. Typically, these reports will have a set schedule and frequency for distribution. The report distribution package must support distribution of these reports without the requirement of manual or user intervention (subsequent to set up and conversion).

5. Online Preview: The report architecture should allow preview of reports online from a user's intelligent workstation prior to actual distribution. Ideally, the report architecture itself would provide support for online preview of reports through software located on the intelligent workstation.

6. Graphical User Interface: The architecture should provide users with a graphical user interface.

7. Bilingual Support: For companies where two or more languages are used, the report architecture must provide a multi-national user interface. (Note that large report runs targeted for multiple users may require the ability to change languages during the report.)

8. Basic Preview Functions: The report architecture should support basic preview functions. These include:

Scrolling up and down.

Scrolling left and right.

Advancing to end or beginning of report without scrolling through intermediate pages.

9. Advanced Preview Functions: In addition to the basic preview functions listed previously, certain advanced preview functions may also be necessary:

Page indexing (allows users to jump to specific report pages).

Section indexing (allows users to jump to specific report sections).

Search capabilities (allows users to search report for occurrence of a specific data stream).

10. Report Level Security: Reports may occasionally contain sensitive information. It is therefore important that access to certain reports be restricted to authorized users. The report architecture should provide a mechanism for implementing report level security. This security must be in place on all platforms with the client/server architecture. At the workgroup level, the security may consist of downloading sensitive report files to a secure directory, and having the LAN administrator release the report as appropriate.

11. Section, Page, and Field Level Security: Defining security at the report section, page, or field level would provide greater flexibility in determining and implementing report security. This is a desirable, though not mandatory, requirement of the report architecture.

12. Background Processing: The report architecture should support the processing of reports in the background while the application works in the foreground during online hours. In other words, processing of reports should not negatively affect online response times, or tie up the user's workstation.

13. Automatic Report Addressing: The report architecture should provide a "humanly intelligible" address for all distributed reports. The address may be used by a print site operator, LAN administrator, or other personnel to manually sort printed output (if required). This criterion can be satisfied by automatic creation of banner pages or other means.

14. Delivery Costing: To provide sufficient information to users to avoid accidentally downloading or printing very large reports during peak usage hours, a distribution costing function can be useful. This function would warn users of reports that would overload the network or a printer. This costing function might provide recipients with a rough estimate of the amount of time that distribution might take. Finally, during the online day, the delivery costing mechanism might disallow transmission of reports that exceed a predetermined cost.

15. Multiple Destinations: The report architecture should support distribution of a single report to single or multiple destinations.

16. Destination Rationalization: For some systems, it is possible that multiple copies of a report will be sent to the

same site—to several different users, for example. In these cases, it is highly desirable to have the report architecture recognize these situations whenever possible and distribute the specified report only once.

17. Automatic Printing: The report architecture should provide automatic print capabilities. Once a report has been distributed for printing (either through a "push" distribution scheduling mechanism or through a "pull" user request) no further user or operations personnel involvement should be necessary to print the report at the specified location.

18. Multiple Print Destinations: The report architecture should support distribution of reports for printing at centralized, remote, or local print sites without user or operations personnel intervention.

19. Variable Printer Types: Printing on multiple types of printers, including line, impact, and laser printers, should be supported. This should not require user intervention—that is, the user should not have to specify the type of target printer. Ideally, the report architecture would default this information from the user's profile or the default printer defined in the local operating system. This criterion requires that the report architecture support several print mechanisms, such as postscript drivers and host/mainframe protocols (for example, Advanced Function Printing [AFP]).

20. Variable Printer Destinations: The report architecture should default the destination printer for.:a specific report (from the user's profile or operating system parameters). Additionally, the architecture should allow the user to change the printer specified. Validation of the print destination also should be included.

21. Special Forms Printing: The report architecture should support distribution of "regular" reports and special forms reports.

22. Font Support: Some reports may be printed on laser printers and/or may support electronic forms text (i.e., including the forms text in the report dataset as opposed to printing the report dataset on a pre-printed form). The architecture should allow multiple fonts to be specified.

23. Report Archival: The report architecture should provide and/or facilitate archival or disposition of report datasets. Ideally, the architecture would permit definition of retention periods and disposition requirements.

24. Report Download: The report architecture should allow distribution of the information contained in a report dataset to a user's intelligent workstation. The information should be in a form that can be imported to a local word processing software, decision support software package, or other appropriate application.

25. Application Transparency: It is desirable for the report architecture to appear to the users as if it were part of the overall application. This does not necessarily mean that the architecture must integrate seamlessly with the application; a message interface between the systems might be acceptable.

26. Selective Printing: It would be desirable for the report architecture to provide users with the ability to print only selected pages or sections of the report. This should reduce paper usage, while still allowing users to obtain a hard copy of the information as required.

27. Print Job Restart: It would be desirable if the report architecture allowed a print job to be restarted from the point of failure rather than having to reprint the entire report. This of particular concern for very large reports.

Technical Criteria

The following is a list of technical criteria that should be considered during the planning for a report architecture:

1. Platform Compatibility: The report architecture must be compatible with the platform architecture. It also should be compatible with local area networks and standalone workstation technology specified in the platform architecture.

2. Wide Area Network Compatibility: Most systems will include support for WAN communication, so the report architecture should be compatible with this environment.

3. Technology Standards: The report architecture should be compliant with existing formal and de facto standards (for example, SQL Database Language, COBOL Programming Language, C Programming Language).

4. External User Directory: The report architecture should make use of an external user directory of preferences and locations.

5. Data Compression in Report Repository: To reduce the storage requirements for the report repository, it is also desirable for the report architecture to support data compression in the repository.

6. Code Page Compatibility: Code page compatibility must be considered when translating characters to ASCII.

Workflow Services **(2890)**

Workflow services control and coordinate the tasks that must be completed in order to process a business event. For example, at XYZ Savings and Loan, in order to receive a promotion, you must complete an essay explaining why you should be promoted. This essay and your personnel file must be routed to numerous individuals who must review the material and approve your promotion. Workflow services coordinate the collection and routing of your essay and your personnel file.

Workflow enables tasks within a business process to be passed among the appropriate participants, in the correct sequence, and facilitates their completion within set times and budgets. Task definition includes the actions required as well as work folders containing forms, documents, images and transactions. It uses business process rules, routing information, role definitions and queues. Workflow functionality is crucial for the customer service and engineering applications to automate the business value chains, and monitor and control the sequence of work electronically.

The business processes can be of a repetitive nature, eg automatically routing and controlling the review of a work plan through the approval stages. These are called production workflows. Conversely it can be an ad hoc process, eg generating and delivering a work order for a special meter reading to a meter reader who is available to perform the task. In production workflows the processes are predefined, whereas ad hoc workflows are created only for a specific nonrecurrent situation. Often it is difficult to determine how much ad hoc functionality that needs to be provided. An overly strict production workflow may not support necessary special cases that must be handled in an ad hoc fasion.

Workflow provides a mechanism to define, monitor and control the sequence of work electronically. These services are typically provided by the server as they often coordinate activities between multiple users on multiple computers.

The following are some of the architectural and integration issues that must be addressed:

Process integration

The workflow system must achieve a seamless integration of multiple processes. The workflow system must control the business process, eg it should be able to open a word processor with the relevant data coming from a previous business process;

Infrastructure integration from PC to mainframe

The ability to interface with the host-based hardware, system software, and database management systems is critical. This is essential because the workflow

SAP - Exhibit 1105
SAP v. Valtrus et al. - IPR2025-00420
Page 183 of 281

system is located between the client-based and host-based processes, ie it can initiate client-based as well as host-based applications;

LAN and WAN connectivity

Connectivity must include all sites for the supported processes, enabling a large number and variety of users to use the workflow system, and thus to execute the business process;

Integration of peripherals

The workflow system should support many different types of printers, modems, fax machines, scanners, and pagers. This is especially important because of the diversity of the users that will be involved, from field crew to managers, each with their own needs and preferences; and

Integration with workflow-participating applications

The key to the efficiency of the workflow system is its capability to integrate with office automation, imaging, electronic mail, and legacy applications.

Workflow can be further divided into the following components:

Role management

Role management ie provides for the assignment of tasks to roles which can then be mapped to individuals.

A role defines responsibilities which are required in completing a business process. A business worker must be able to route documents and folders to a role, independent of the specific person, or process filling that role. For example, a request is routed to a supervisor role or to Purchasing, rather than to "Mary" or "Tom." If objects are routed to Mary and Mary leaves the company or is reassigned, a new recipient under a new condition would have to be added to an old event. Roles are also important when a number of different people have the authority to do the same work, such as claims adjusters; just assign the request to the next available person. In addition, a process or agent can assume a role; it doesn't need to be a person. Role Management Services provide this additional level of directory indirection.

Route management

Route management enables the routing of tasks to the next role, which can be done in the following ways:

Serial—the tasks are sequentially performed;

Parallel—the work is divided among different players;

Conditional—routing is based upon certain conditions; and

Ad hoc—work which is not part of a predefined process.

Workflow routing services route "work" to the appropriate workflow queues. When an application completes processing a task, it uses these services to route the work-in-progress to the next required task or tasks and, in some cases, notify interested parties of the resulting work queue changes.

The automatic movement of information and control from one workflow step to another requires work profiles that describe the task relationships for completing various business processes. The concept of Integrated Performance Support can be exhibited by providing user access to these work profiles. Such access can be solely informational—to allow the user to understand the relationship between tasks, or identify which tasks need to be completed for a particular work flow—or navigational—to allow the user to move between tasks.

Route Management Services also support the routing and delivery of necessary information (e.g., documents, data, forms, applications, etc.) to the next step in the work flow as needed.

Rule Management

A business process workflow is typically composed of many different roles and routes. Decisions must be made as to what to route to which role, and when. Rule Management Services support the routing of workflow activities by providing the intelligence necessary to determine which routes are appropriate given the state of a given process and knowledge of the organization's workflow processing rules. Rule Management Services are typically implemented through easily maintainable tables or rule bases which define the possible flows for a business event.

Queue Management

These services provide access to the workflow queues which are used to schedule work. In order to perform workload analysis or to create "to do lists" for users, an application may query these queues based on various criteria (a business event, status, assigned user, etc.). In addition, manipulation services are provided to allow queue entries to be modified.

Workflow services allow users and management to monitor and access workflow queue information and to invoke applications directly.

Is There a Need for Reporting and Management Facilities?

Typical workflow application requirements are better general management control and better management of change. Proactive system action, audit trails and system administration features like work queue reporting are important administration tools. Some of the areas for monitoring for improvement are employee productivity, process performance, and forecasting/scheduling. Where any form of customer service is involved, features like status reports on individual cases can sharpen customer response times while performance monitoring of groups and individuals can help quality improvement and efficiency exercises. Note that reports and reporting does not necessarily mean paper reports that are distributed in a traditional manner, it can mean electronic messages or even triggers based on specific events.

Are Cooperative Applications Present?

Workflow management is frequently required in cooperative applications because the users are generally professionals, the flow of work in the organization is frequently highly variable, the application units of work (legal case, sales order) are processed for long periods of elapsed time, and work often moves from one processing site to another. As data and application logic are split, better control is needed to track processing/data status across location.

Will There be Business Process Re-engineering?

Workflow is a logical complement to BPR and the trend is moving toward using workflow software to re-engineer new business processes on a workgroup or project basis.

Is the Business Process Well Defined?

If rules or conditions can be identified which define the business process, with few exception conditions, workflow tools can then automate areas such as information routing, task processing, and work-in-process reporting.

Are Fixed Delays or Deadlines Involved?

Workflow has been used to regulate delays and deadlines such as those associated with government regulations, con-