US 20020156756A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2002/0156756 A1**

Stanley et al. (43) **Pub. Date: Oct. 24, 2002**

(54) INTELLIGENT MOLECULAR OBJECT DATA STRUCTURE AND METHOD FOR APPLICATION IN HETEROGENEOUS DATA ENVIRONMENTS WITH HIGH DATA DENSITY AND DYNAMIC APPLICATION NEEDS

(75) Inventors: **Robert A. Stanley**, Emeryville, CA (US); **Erich A. Gombocz**, San Francisco, CA (US)

Correspondence Address:
FLEHR HOHBACH TEST ALBRITTON & HERBERT LLP
Suite 3400
Four Embarcadero Center
San Francisco, CA 94111-4187 (US)

(73) Assignee: **Biosentients, Inc.**

(21) Appl. No.: **10/010,724**

(22) Filed: **Dec. 6, 2001**

Related U.S. Application Data

(60) Provisional application No. 60/254,063, filed on Dec. 6, 2000. Provisional application No. 60/254,062, filed on Dec. 6, 2000. Provisional application No. 60/254,064, filed on Dec. 6, 2000. Provisional application No. 60/259,050, filed on Dec. 29, 2000. Provisional application No. 60/264,238, filed on Jan. 25, 2001. Provisional application No. 60/266,957, filed on Feb. 6, 2001. Provisional application No. 60/276,711, filed on Mar. 16, 2001. Provisional application No. 60/282,656, filed on Apr. 9, 2001. Provisional application No. 60/282,658, filed on Apr. 9, 2001. Provisional application No. 60/282,654, filed on Apr. 9, 2001. Provisional application No. 60/282,657, filed on Apr. 9, 2001. Provisional application No. 60/282,655, filed on Apr. 9, 2001. Provisional application No. 60/282,979, filed on Apr. 10, 2001. Provisional application No. 60/282,989, filed on Apr. 10, 2001. Provisional application No. 60/282,991, filed on Apr. 10, 2001. Provisional application No. 60/282,990, filed on Apr. 10, 2001.

Publication Classification

(51) **Int. Cl.**^{7} **G06N 5/02**

(52) **U.S. Cl.** **706/47**

(57) ABSTRACT

Methods are provided to define and describe the creation, architecture and embodiment of Intelligent Molecular Object data ("IMO", hereinafter "Intelligent Object"). The Intelligent Object is a software product invented to enable unified user presentation, accessing, routing and functionally integrated processing of potentially (but not necessarily) diverse data. Each Intelligent Object provides a unified, interactive user interface to uniquely identified, functionally integrated data from potentially heterogeneous, data content from potentially diverse database back-ends and/or data resources. Said data content includes but is not limited to data such as gene sequence data, protein expression data, 2D gel electrophoresis data, chemical structure data, bio-assay data, image data, text data, audio data or other data from an extensible variety of data types and structures. Intelligent Object technology improves data usability and rate of access to query-relevant elements, attributes and other meta-data, and provides means for functional multidimensional analysis. The technology provides real-time integrated access to previously incompatible data, including synchronized access to off-line and/or latent response data and significantly reduces response time for queries of large datasets. The technology provides data management and access across diverse hardware and software platforms and research applications. The technology secures data for global network use and exchange, and provides extensible options, including ownership management, data integrity, use-tracking, and selective access. Additionally, Intelligent Object handling and storage technology for customization, analysis, and exchange is provided. Methods including software processes, sets of instructions, procedural rules, and look-up tables are provided to define and describe: Intelligent Object creation; unified functional presentation; unique data object, content, user and session identification; user and session authentication and permission for data access; dynamic root data and meta-data routing of multidimensional, vectorized content information; data logging, auditing, status management and validation state alerting; raw data matrix and matrix structure definition; meta-data indexing and query optimization; functional content and attribute definition for database or application access and routing for said database or application; direct data-to-data information interchange; graphical data viewing and analysis; and text annotation integration for diversified data in networked Life Sciences applications environments.

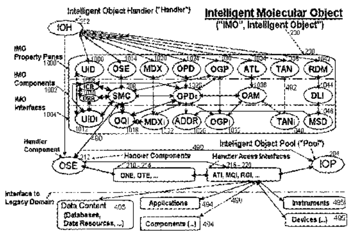
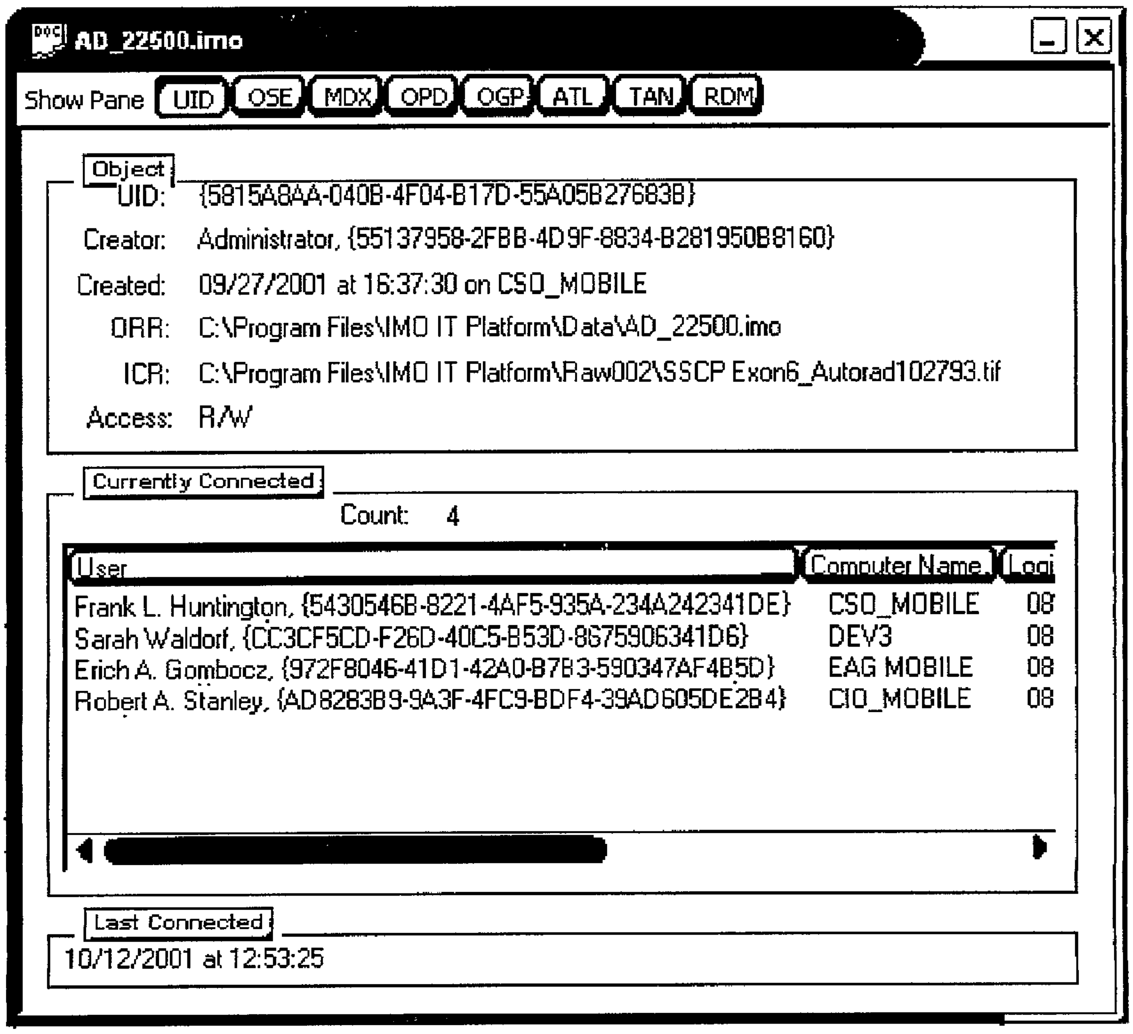


FIG. 1



Object

UID: {5815A8AA-040B-4F04-B17D-55A05B27683B}
Creator: Administrator, {55137958-2FBB-4D9F-8834-B281950B8160}
Created: 09/27/2001 at 16:37:30 on CSO_MOBILE
ORR: C:\Program Files\IMO IT Platform\Data\AD_22500.imo
ICR: C:\Program Files\IMO IT Platform\Raw002\SSCP Exon6_Autorad102793.tif
Access: R/W

Currently Connected

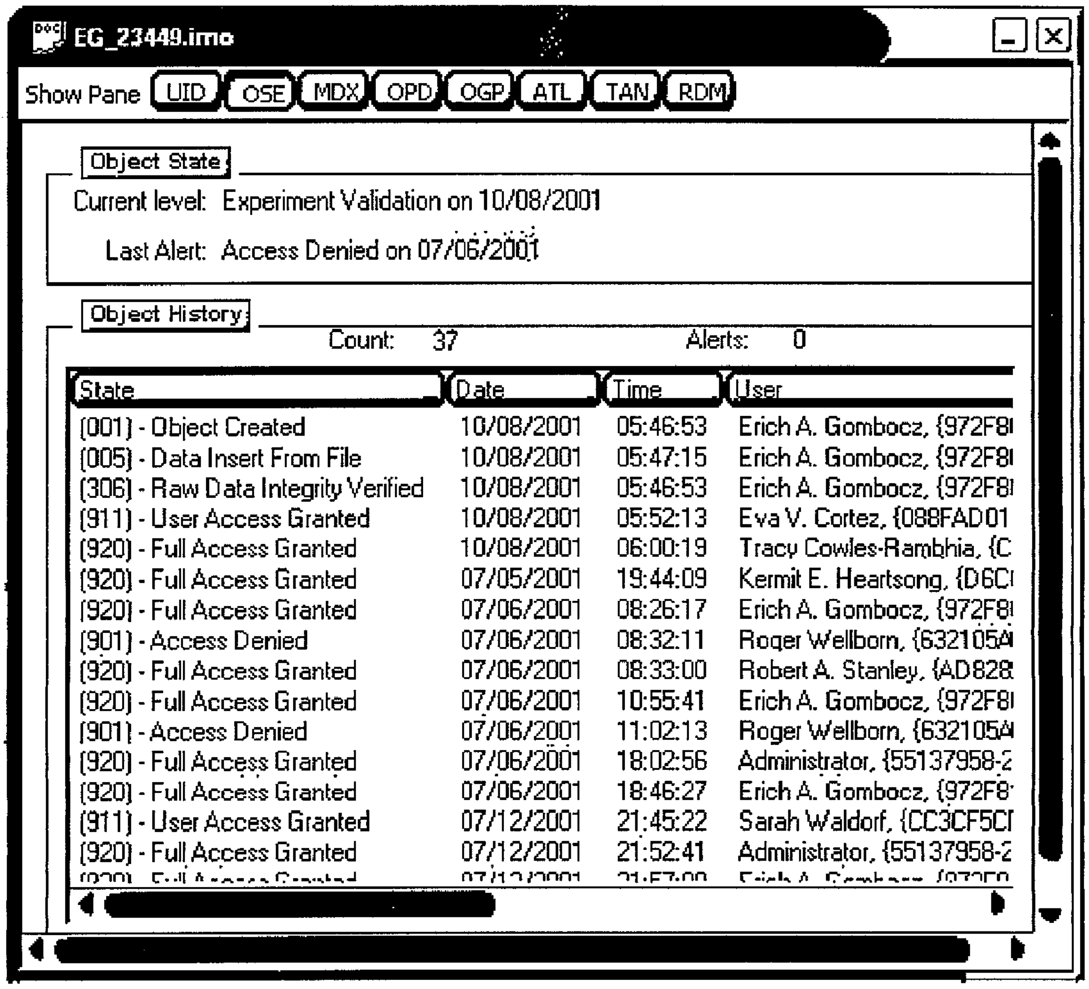
Count: 4

User	Computer Name	Logi
Frank L. Huntington, {5430546B-8221-4AF5-935A-234A242341DE}	CSO_MOBILE	08
Sarah Waldorf, {CC3CF5CD-F26D-40C5-B53D-8675906341D6}	DEV3	08
Erich A. Gombocz, {972F8046-41D1-42A0-B7B3-590347AF4B5D}	EAG MOBILE	08
Robert A. Stanley, {AD8283B9-9A3F-4FC9-BDF4-39AD605DE2B4}	CIO_MOBILE	08

Last Connected

10/12/2001 at 12:53:25

FIG. 2



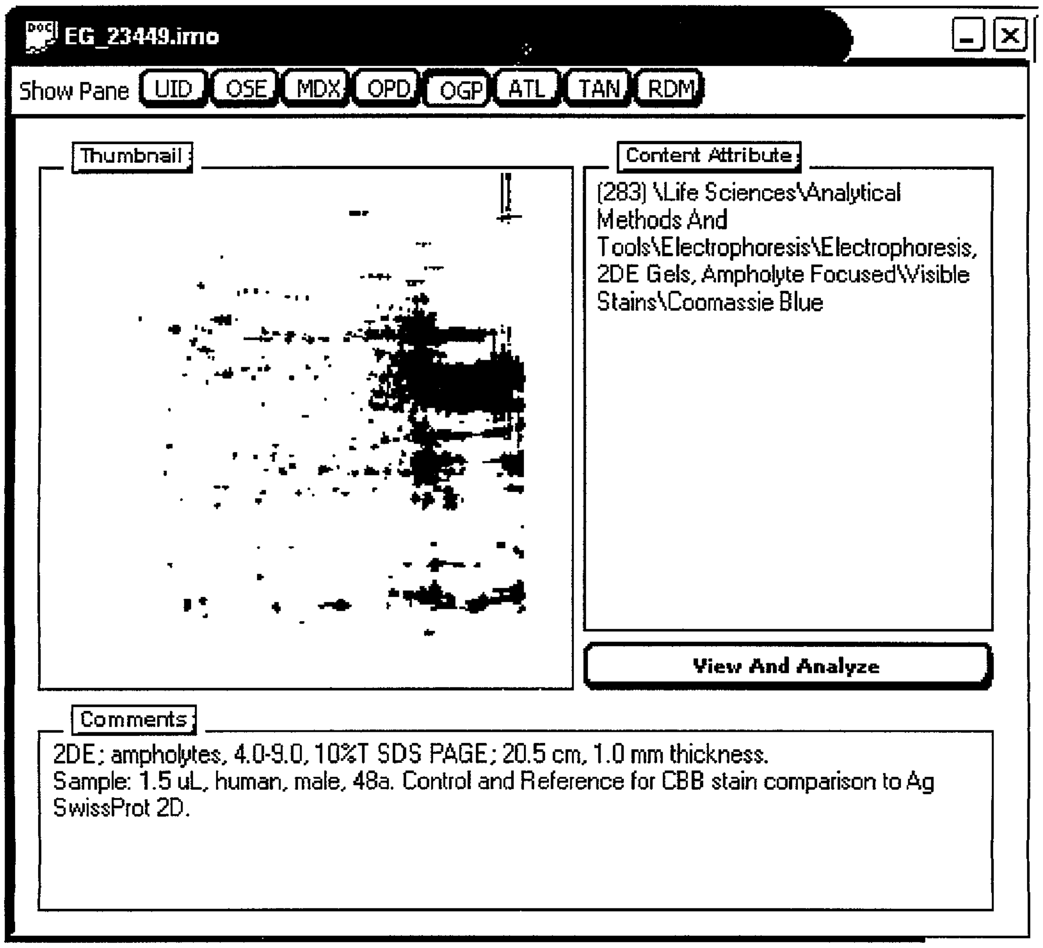
Current level: Experiment Validation on 10/08/2001

Last Alert: Access Denied on 07/06/2001

Count: 37 Alerts: 0

State	Date	Time	User
[001] - Object Created	10/08/2001	05:46:53	Erich A. Gombocz, {972F8
[005] - Data Insert From File	10/08/2001	05:47:15	Erich A. Gombocz, {972F8
[306] - Raw Data Integrity Verified	10/08/2001	05:46:53	Erich A. Gombocz, {972F8
[911] - User Access Granted	10/08/2001	05:52:13	Eva V. Cortez, {088FAD01
[920] - Full Access Granted	10/08/2001	06:00:19	Tracy Cowles-Rambhia, {C
[920] - Full Access Granted	07/05/2001	19:44:09	Kermit E. Heartsong, {D6C
[920] - Full Access Granted	07/06/2001	08:26:17	Erich A. Gombocz, {972F8
[901] - Access Denied	07/06/2001	08:32:11	Roger Wellborn, {632105A
[920] - Full Access Granted	07/06/2001	08:33:00	Robert A. Stanley, {AD828
[920] - Full Access Granted	07/06/2001	10:55:41	Erich A. Gombocz, {972F8
[901] - Access Denied	07/06/2001	11:02:13	Roger Wellborn, {632105A
[920] - Full Access Granted	07/06/2001	18:02:56	Administrator, {55137958-2
[920] - Full Access Granted	07/06/2001	18:46:27	Erich A. Gombocz, {972F8
[911] - User Access Granted	07/12/2001	21:45:22	Sarah Waldorf, {CC3CF5C
[920] - Full Access Granted	07/12/2001	21:52:41	Administrator, {55137958-2

FIG. 3:



(283) \Life Sciences\Analytical Methods And Tools\Electrophoresis\Electrophoresis, 2DE Gels, Ampholyte Focused\Visible Stains\Coomassie Blue

2DE; ampholytes, 4.0-9.0, 10%T SDS PAGE; 20.5 cm, 1.0 mm thickness.
Sample: 1.5 uL, human, male, 48a. Control and Reference for CBB stain comparison to Ag SwissProt 2D.

FIG. 4:

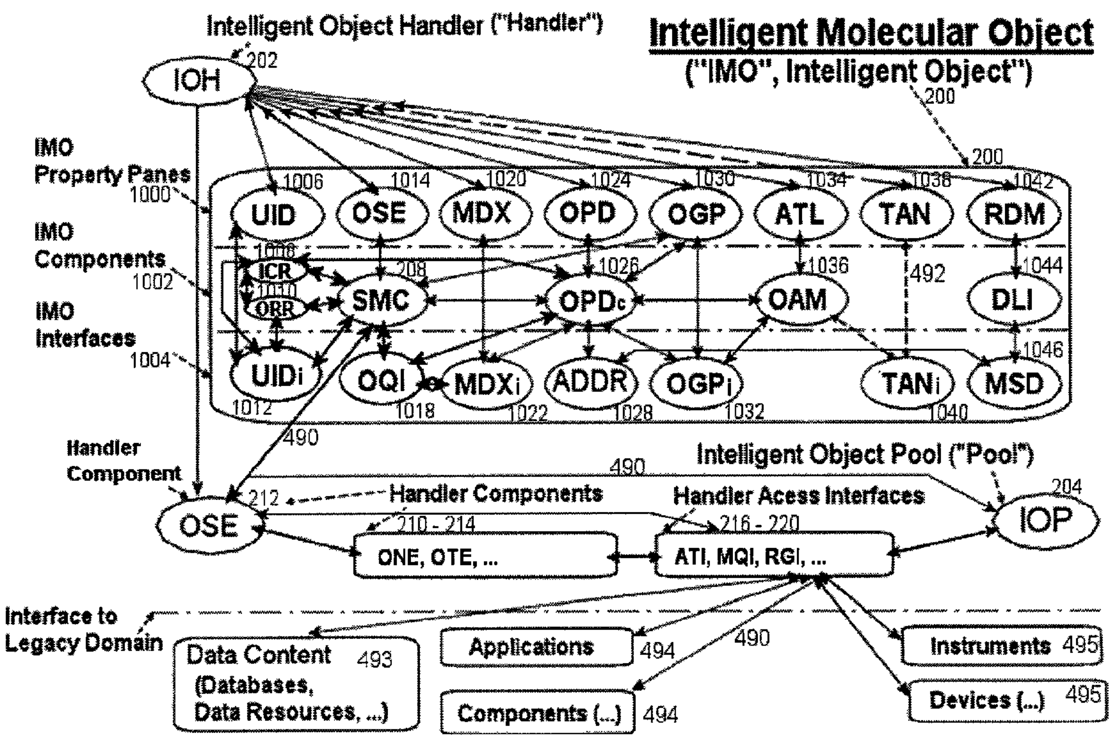

Intelligent Object Handler ("Handler")
202
IOH
Intelligent Molecular Object
("IMO", Intelligent Object")
200
200
IMO Property Panes
1000
IMO Components
1002
IMO Interfaces
1004
UID 1006
OSE 1014
MDX 1020
OPD 1024
OGP 1030
ATL 1034
TAN 1038
RDM 1042
ICR 1008
ORR 1010
SMC 208
OPDc 1026
OAM 1036
492
DLI 1044
UIDi 1012
OQI 1018
MDXi 1022
ADDR 1028
OGPi 1032
TANi 1040
MSD 1046
490
Handler Component
OSE 212
Handler Components
210 - 214
Intelligent Object Pool ("Pool")
204
IOP
Handler Acess Interfaces
216 - 220
ONE, OTE, ...
ATI, MQI, RGI, ...
Interface to Legacy Domain
Data Content (Databases, Data Resources, ...) 493
Applications 494
Components (...) 494
Instruments 495
Devices (...) 495

FIG. 5

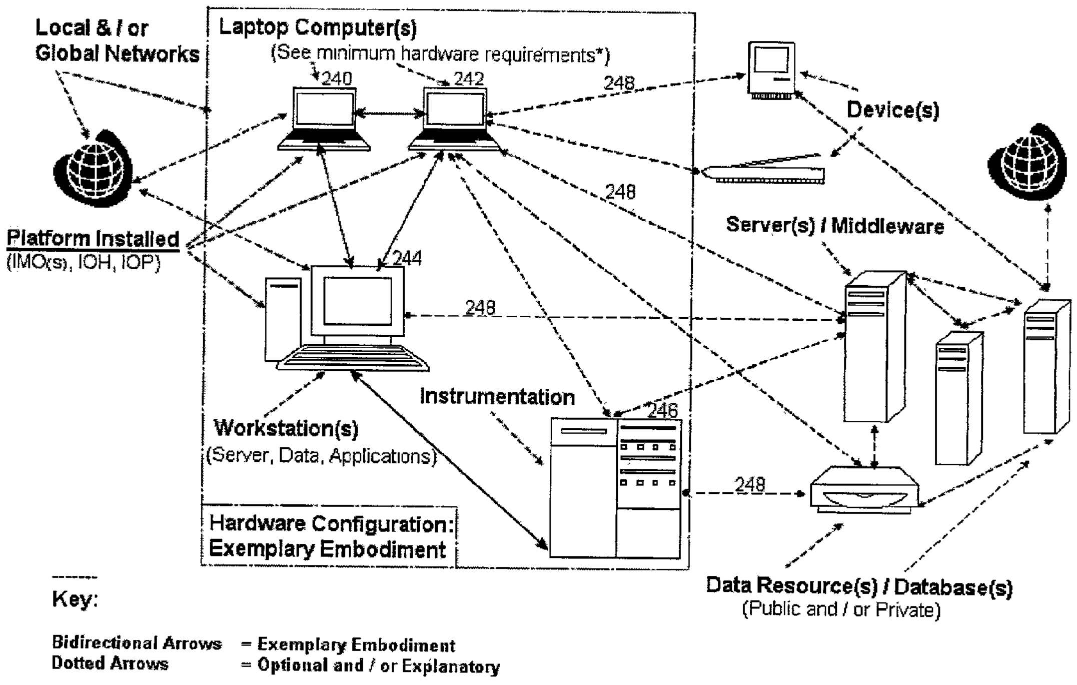
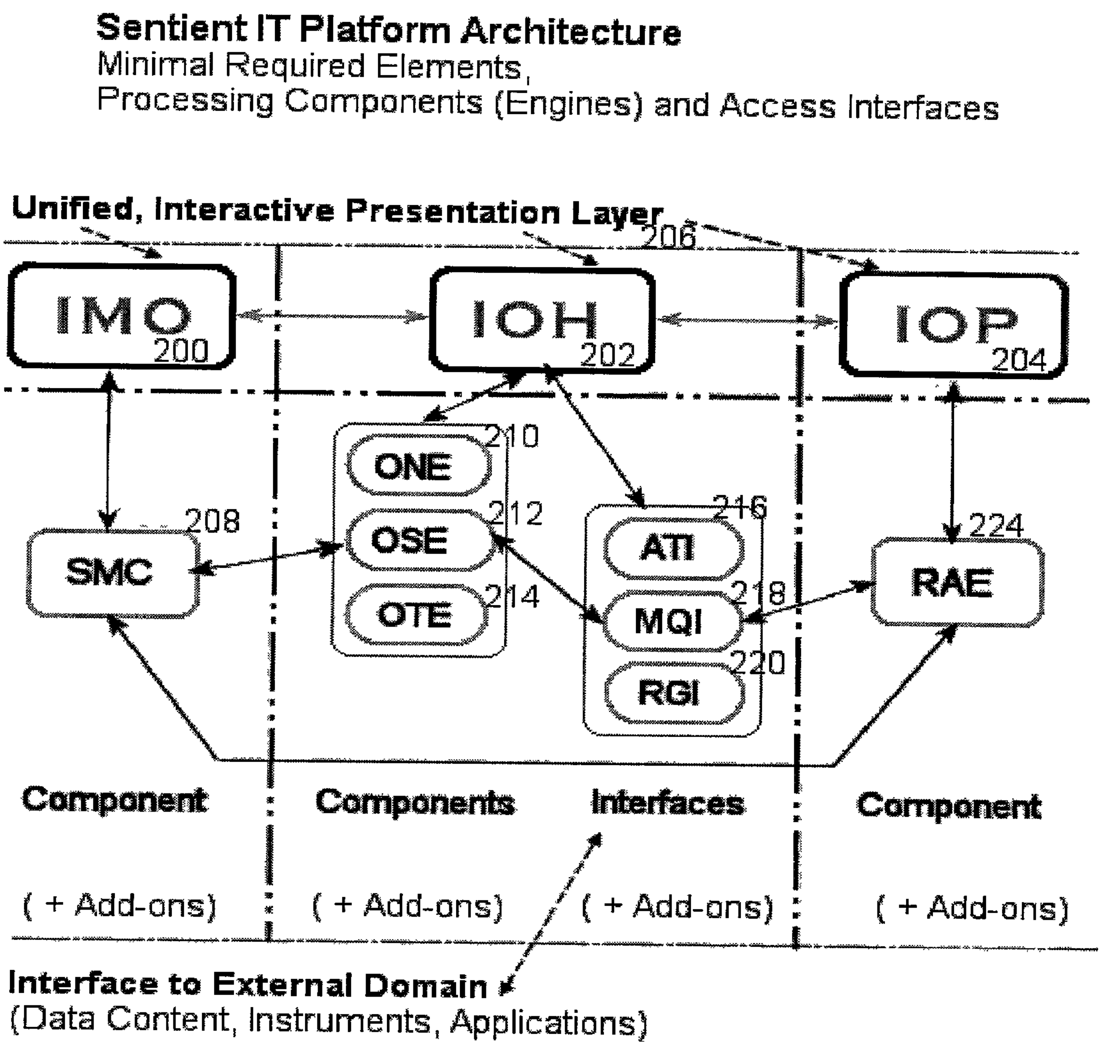

Local & / or Global Networks
Laptop Computer(s)
(See minimum hardware requirements*)
240
242
248
Device(s)
248
Server(s) / Middleware
Platform Installed
(IMO(s), IOH, IOP)
244
248
Instrumentation
246
Workstation(s)
(Server, Data, Applications)
248
Hardware Configuration: Exemplary Embodiment
Data Resource(s) / Database(s)
(Public and / or Private)
Key:
Bidirectional Arrows = Exemplary Embodiment
Dotted Arrows = Optional and / or Explanatory

FIG. 6


Sentient IT Platform Architecture
Minimal Required Elements,
Processing Components (Engines) and Access Interfaces
Unified, Interactive Presentation Layer
206
IMO
200
IOH
202
IOP
204
SMC
208
ONE
210
OSE
212
OTE
214
ATI
216
MQI
218
RGI
220
RAE
224
Component
Components
Interfaces
Component
(+ Add-ons)
(+ Add-ons)
(+ Add-ons)
(+ Add-ons)
Interface to External Domain
(Data Content, Instruments, Applications)

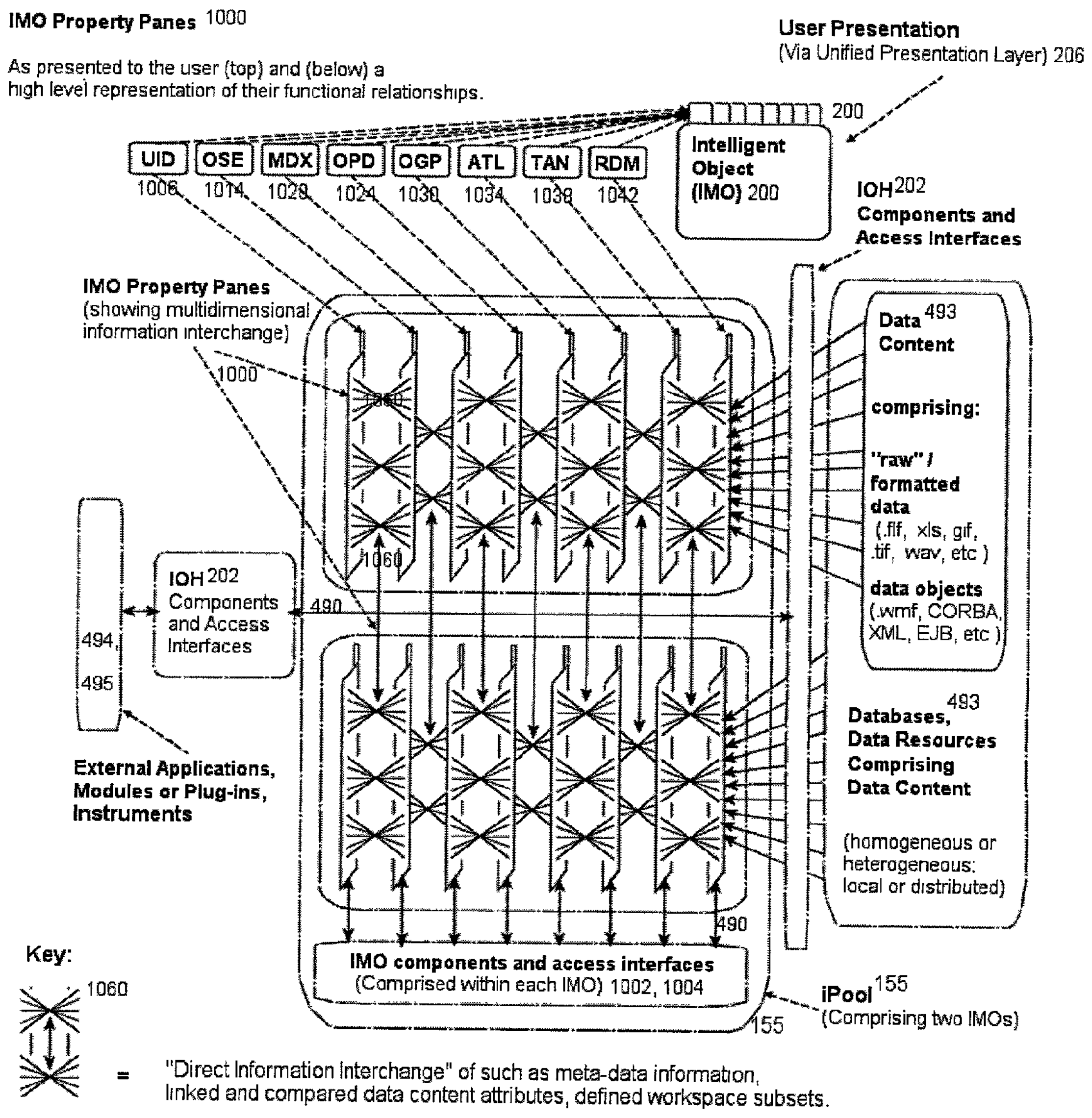

IMO Property Panes 1000
As presented to the user (top) and (below) a high level representation of their functional relationships.
User Presentation
(Via Unified Presentation Layer) 206
200
UID
OSE
MDX
OPD
OGP
ATL
TAN
RDM
1006
1014
1020
1024
1030
1034
1038
1042
Intelligent Object (IMO) 200
IOH 202 Components and Access Interfaces
IMO Property Panes (showing multidimensional information interchange)
1000
1060
Data 493 Content
comprising:
"raw" / formatted data (.flf, xls, gif, .tif, wav, etc)
data objects (.wmf, CORBA, XML, EJB, etc)
IOH 202 Components and Access Interfaces
490
494, 495
External Applications, Modules or Plug-ins, Instruments
Databases, 493 Data Resources Comprising Data Content
(homogeneous or heterogeneous; local or distributed)
490
IMO components and access interfaces (Comprised within each IMO) 1002, 1004
iPool 155 (Comprising two IMOs)
155
Key:
1060
= "Direct Information Interchange" of such as meta-data information, linked and compared data content attributes, defined workspace subsets.

FIG. 7

INTELLIGENT MOLECULAR OBJECT DATA STRUCTURE AND METHOD FOR APPLICATION IN HETEROGENEOUS DATA ENVIRONMENTS WITH HIGH DATA DENSITY AND DYNAMIC APPLICATION NEEDS

RELATED APPLICATIONS

[0001] Priority is hereby claimed under 35 U.S.C. 120 and/or 35 U.S.C. 119(e) to the following United States Provisional and Utility Patent Applications, each of which is hereby incorporated by reference:

[0002] U.S. Utility patent application Ser. No. ______ (Attorney Docket No. A-70134/RMA) filed Dec. 6, 2001 and entitled Data Pool Architecture, System, And Method For Intelligent Object Data In Heterogeneous Data Environments;

[0003] U.S. Utility patent application Ser. No. ______ (Attorney Docket No. A-70135/RMA) filed Dec. 6, 2001 and entitled Intelligent Molecular Object Data Structure and Method for Application in Heterogeneous Data Environments with High Data Density and Dynamic Application Needs;

[0004] U.S. Utility patent application Ser. No. ______ (Attorney Docket No. A-70136/RMA) filed Dec. 6, 2001 and entitled Intelligent Object Handling Device and Method for Intelligent Object Data in Heterogeneous Data Environments with High Data Density and Dynamic Application Needs;

[0005] U.S. Utility patent application Ser. No. ______ (Attorney Docket No. A-70310/RMA) filed Dec. 6, 2001 and entitled System, Method, Software Architecture, And Business Model For An Intelligent Object Based Information Technology Platform;

[0006] U.S. Provisional Application Serial No. 60/254,063 filed Dec. 6, 2000 entitled Data Pool Architecture for Intelligent Molecular Object Data in Heterogeneous Data Environments with High Data Density and Dynamic Application Needs;

[0007] U.S. Provisional Application Serial No. 60/254,062 filed Dec. 6, 2000 entitled Intelligent Molecular Object Data for Heterogeneous Data Environments with High Data Density and Dynamic Application Needs;

[0008] U.S. Provisional Application Serial No. 60/254,064 filed Dec. 6, 2000 entitled Handling Device for Intelligent Molecular Object Data in Heterogeneous Data Environments with High Data Density and Dynamic Application Needs;

[0009] U.S. Provisional Application Serial No. 60/259,050 filed Dec. 29, 2000 entitled Object State Engine for Intelligent Molecular Object Data Technology;

[0010] U.S. Provisional Application Serial No. 60/264,238 filed Jan 25, 2001 entitled Object Translation Engine Interface For Intelligent Molecular Object Data;

[0011] U.S. Provisional Application Serial No. 60/266,957 filed Feb. 6, 2001 entitled System, Method, Software Architecture and Business Model for an Intelligent Molecular Object Based Information Technology Platform;

[0012] U.S. Provisional Application Serial No. 60/276,711 filed Mar. 16, 2001 entitled Application Translation Interface For Intelligent Molecular Object Data In Heterogeneous Data Environments With Dynamic Application Needs;

[0013] U.S. Provisional Application Serial No. 60/282,656 filed Apr. 9, 2001 entitled Result Generation Interface For Intelligent Molecular Object Data In Heterogeneous Data Environments With Dynamic Application Needs;

[0014] U.S. Provisional Application Serial No. 60/282,658 filed Apr. 9, 2001 entitled Knowledge Extraction Engine For Intelligent Object Data In Heterogeneous Data Environments With Dynamic Application Needs;

[0015] U.S. Provisional Application Serial No. 60/282,654 filed Apr. 9, 2001 entitled Result Aggregation Engine For Intelligent Object Data In Heterogeneous Data Environments With Dynamic Application Needs;

[0016] U.S. Provisional Application Serial No. 60/282,657 filed Apr. 9, 2001 entitled Automated Applications Assembly Within Intelligent Object Data Architecture For Heterogeneous Data Environments With Dynamic Application Needs;

[0017] U.S. Provisional Application Serial No. 60/282,655 filed Apr. 9, 2001 entitled System, Method And Business Model For Productivity In Heterogeneous Data Environments;

[0018] U.S. Provisional Application Serial No. 60/282,979 filed Apr. 10, 2001 entitled Legacy Synchronization Interface For Intelligent Molecular Object Data In Heterogeneous Data Environments With Dynamic Application Needs;

[0019] U.S. Provisional Application Serial No. 60/282,989 filed Apr. 10, 2001 entitled Object Query Interface For Intelligent Molecular Object Data In Heterogeneous Data Environments With Dynamic Application Needs;

[0020] U.S. Provisional Application Serial No. 60/282,991 filed Apr. 10, 2001 entitled Distributed Learning Engine For Intelligent Molecular Object Data In Heterogeneous Data Environments With Dynamic Application Needs; and

[0021] U.S. Provisional Application Serial No. 60/282,990 filed Apr. 10, 2001 entitled Object Normalization For Intelligent Molecular Object Data In Heterogeneous Data Environments With Dynamic Application Needs;

[0022] each of which U.S. utility and U.S. provisional patent application is hereby incorporated by reference in its entirety.

FIELD OF INVENTION

[0023] This invention pertains generally to system, method, computer program product, data structure and architecture, data management, and software architecture; and more particularly to system, method, computer program product, and data structure and architecture, data management, and software architecture in the life sciences, biotechnology, therapeutic diagnostic and intervention, pharmaceuticals, and bioinformatics.

BACKGROUND

[**0024**] As demand for effective Information Technology (IT) software to provide global data access and integrated scientific and business solutions has grown, significant challenges have become evident. A central problem poses access, integration, and utilization of large amounts of new and valuable information generated in each of the major industries. Lack of unified, global, real-time data access and analysis is detrimental to crucial business processes, which include new product discovery, product development, decision-making, product testing and validation, and product time-to-market. Additionally, the importance of functionally integrating multiple dimensions of heterogeneous data in the field, such as protein expression data, chemical structure data, bioassay data and clinical text data, is recognized (Lin, D., et al., 2001).

[**0025**] With the completion of the sequence of the human genome and the continued effort in understanding protein expression in the life sciences, a wealth of new genes are being discovered that will have potential as targets for therapeutic intervention. As a result of this new information, however, Biotech and Pharmaceutical companies are drowning in a flood of data. In the Life Sciences alone, approximately 1 Terabyte of data is generated per company and day, of which currently the vast majority is unutilized for several reasons.

[**0026**] Data are contained in diversified system environments using different formats, heterogeneous databases and have been analyzed using different applications. These applications may each apply different processing to those data. Competitive software, based on proprietary platforms for network and applications analysis, have utilized data platform technologies such as SQL with open database connectivity (ODBC), component object model (COM), Object Linking and Embedding (OLE) and/or proprietary applications for analysis as evidenced in patents from such companies as Sybase, Kodak, IBM, and Cellomics in U.S. Pat. No. 6,161,148, No. 6,132,969, No. 5,989,835, No. 5,784,294, for data management and analysis, each of which patents are hereby incorporated by reference. Because of this diversity, despite the fact that the seamless integration of public, legacy and new data is crucial to efficient drug discovery and life science research, current data mining tools cannot handle all data and analyze their functional relationships simultaneously. There is a significant lack of data handling methods, which can utilize these data in a secure, manageable way. The shortcomings of these technologies are evident within heterogeneous software and hardware environments with global data resources. Despite the fact that the seamless integration of public, legacy and new data is crucial to efficient research (particularly in the life sciences), product discovery (such as for example drug, or treatment regime discovery) and distribution, current data mining tools cannot handle or validate all diverse data simultaneously.

[**0027**] With the expansion of high numbers of dense data in a global environment, user queries often require costly massive parallel or other supercomputer-oriented processing in the form of mainframe computers and/or cluster servers with various types of network integration software pieced together for translation and access functionality as evidenced by such companies as NetGenics, IBM and ChannelPoint in U.S. Pat. No. 6,125,383 No. 6,078,924, No. 6,141,660, No. 6,148,298, each of which patents are herein incorporated by reference—(e.g. Java, CORBA, "wrapping", XML) and networked supercomputing hardware as evidenced by such companies as IBM, Compaq and others in patents such as for example U.S. Pat. Nos. 6,041,398, 5,842,031, each of which is hereby incorporated by reference. Even with these expensive software and hardware infrastructures, significant time-delays in result generation remain the norm.

[**0028**] In part due to the flood of data and for other reasons as well, there is a significant redundancy within the data, making queries more time consuming and less efficient in their results. Tools are not yet in place which can effectively detect data redundancy over heterogeneous data types and network environments, especially of data content subsets within data files, and provide ranked and validated multiple addressing and/or removal of said redundant data. The flood of new and legacy data results in a significant redundancy within the data making queries more time consuming and less efficient in their results.

[**0029**] With the advent of distinct differentiations in the field of genomics, proteomics, bioinformatics and the need for informed decision making in the life sciences, the state of object data is crucial for their overall validation and weight in complex, multi-disciplinary queries. This is even more important due to inter-dependencies of a variety of data at different states. Furthermore, because biological data describe a "snapshot" representing a unique moment of complex processes at a defined state of the organism, data obtained at any time refer to this unique phase of metabolism. Thus, in order to account for meaningful comparison, only data in similar states can be utilized. Therefore, there is a growing need for an object data state processing engine, which allows to continuously monitor, govern, validate and update the data state based on any activities of intelligent molecular objects in real-time. Currently, these capabilities are not broadly available for network data structures, and they are not available for data structures integrating heterogeneous data over distributed network environments.

[**0030**] With the advent of distinct differentiations in the field of genomics, proteomics, bioinformatics and the need for informed decision making in the life sciences, access to all data is crucial for overall validation and weight in complex, multi-disciplinary queries. This is even more important due to inter-dependencies of a variety of data at different states. The current individual data translation approach does not support these needs. Most of these problems require real-time processing; automated, instant data translation of data from different sources; and integration of heterogeneous applications and analytical components for their solutions. Data contained in diversified system environments may use different formats, heterogeneous databases and different applications, each of which may apply different processing to those data. Therefore, there is a growing and unmet need for an automated object data translation engine, which allows for bi-directional translation of multidimensional data from various sources into intelligent molecular objects in real-time. Currently, data translation processes between different data types are time-consuming and require administrative exchange of information on data structures application programming interfaces (API's) and other dependencies, as required by the latest technologies such as Incellico's CELL, IBM's DiscoveryLink, Netgenic's Synergy and Tripos' MetaLayer solutions (Haas et al 2001). These processes, although available and used, have a number of shortcomings. Despite the fact that the rapid seamless integration of public, legacy and newly emerging data is crucial to efficient drug discovery and life science research, unique "wrappers" or translation layers must currently be designed and programmed in order to translate each of those data sets correctly, and even with this manual integration, multiple data types and dimensions of data interdependencies are not made available, or "functionally integrated" for detailed qualitative and quantitative comparison and analysis across data types and dimensions. These solutions currently require significant effort and resources in both, software development and data processing, and the need for improvements such as those offered by this invention are recognized.

[0031] An additional consideration, which is prohibitive to change towards a more homogeneous infrastructure is the missing of fluently definable object representation definition protocols to prepare and present data objects for unified, functionally integrated interaction within heterogeneous environments. There is a lack of defined sets of user interaction and environment definition protocols needed to provide means for intelligent data mining and optimization of multidimensional analysis to achieve validated solutions. Data currently are interacted with and presented in diverse user interfaces with dedicated, unique features and protocols, preventing universal, unified user access. Thus, a homogeneous, unified presentation such as a flexibly network-enabled graphical user interface, which integrates components from diverse applications and laboratory systems environments over a variety of connections and protocols, is highly desirable, but currently non-existent for real-time data access and analysis utilizing diverse applications and data.

[0032] Finally, an additional consideration, which is prohibitive to change towards a homogenous data and applications infrastructure, is cost. The cost to bring legacy systems up to date, to retool a company's intranet-based software systems, to create a unified environment utilizing existing software products and tools such as CORBA, JAVA, XML, SQL and classic data warehousing techniques, can be time-consuming and expensive. Conventional practices require retooling and/or translating at both application and hardware layers, as evidenced by such companies as Unisys and IBM in U.S. Pat. Nos. 6,038,393, 5,634,015, and may be prohibitively expensive for smaller and medium-sized companies or groups wishing to access this type of functionality.

[0033] Because of the constraints outlined above, it is nearly impossible to extract useful, functionally integrated information from the entity of data within reasonable computing time and efforts. For these reasons, the development of comprising a unique architecture and system, comprising a unique application framework, data structure, and database structure, is unavailable and needed to overcome these obstacles (Hobbs, D. W. 2001).

LITERATURE

[0034] Andreoli, J-M., In: Agha G., Wegner P., Yonezawa A.(eds.): Research Directions in Concurrent Object-Oriented Programming, MIT Press (1993): 260-263; Bertino E., Urban S., Rundensteiner E. A. (eds.): Theory and Practice of Object Systems (1999) 5 (3): 125-197; Chaudhri A. B., McCann J. A., Osmon P.: Theory and Practice of Object Systems (1999) 5 (4): 263-279; Cai D., McTear M. F., McClean S. I.: International Journal of Intelligent Systems (2000): 15 (8): 745-761; Hert C. A., Jacob E. K., Dawson P.: Journal of the American Society for Information Science (2000) 51 (11): 971-988; Hobbs, D. W., Chemical and Engineering News. (2001) 79 (13): 266; Lin, D., et al.: American Genomic/Proteomic Technology (2001) 1 (1): 38-46; Williams R. J., In: Miller W., Thomas I., Sutton R. S., Werbos, P. J. (eds.): Neural Networks for Control, MIT Press (1990): 97-114; Wilson G. V., Lu P. (eds.): Parallel Programming and C++, MIT Press (1996): 257-280.; C. N. Lauro, G. Giordano, R. Verde: Applied Stochastic Models and Data Analysis: A multidimensional approach to conjoint analysis (1998) 14 (4): 265-274; Meyer, Bertrand: IEEE Computer. (1999) 32 (1): 139-140.; Chalmers, Mathew: Journal of the American Society for Information Science. (1999) 50 (12): 1108-1118.; Teasley, Stephanie and Steven Wolinsky: Science. (Jun. 22, 2001) 292:2254; Haas, L. M., et al: IBM Systems Journal. (2001) 40 (2): 489-511.; Siepel, A., et al: IBM Systems Journal. (2001) 40 (2): 570-591; and Steiner, S. and Witzmann, F. A. Electrophoresis. (2000) 21: 2099-2104; each of which publications are incorporated by reference.

[0035] The following United States Patents: U.S. Pat. No. 5,596,744, No. 5,867,799, No. 5,745,895, No. 6,076,088, No. 5,706,453, No. 5,767,854, No. 6,035,300, No. 6,145,009, No. 5,664,066, No. 5,862,325, No. 6,016,495, No. 6,119,126, No. 6,088,717, No. 6,052,722, No. 6,064,382, No. 6,134,581, and No. 6,146,027; each of which are incorporated by reference.

SUMMARY

[0036] The invention provides structure, method, computer program and computer program product for an intelligent object. An Intelligent Object, advantageously enabled in computer program software code or instructions, provides methods for: data and user identification and status management; functional integration of, and access to, potentially diverse data over a variety of computing infrastructures; integration of multiple data types and dimensions for efficient and accurate multidimensional, parallel queries and analyses; for diversified data content and dynamic applications needs in heterogeneous local and/or networked computing and applications environments, especially in Life Sciences computing and applications environments.

[0037] Methods for creation of Intelligent Objects are provided which, upon user initiation, queries, data acquisition protocols or data import requests invoke the unique object identifier property pane through a unified functional presentation layer. The unique object identifier property pane assigns each new data object a globally unique identification upon creation and generates a minimum set of functional property panes within the object, which account for unified viewing and processing. Once the object's state recording is started, active identification for all connections to and activities on the Intelligent Object are listed within the unique object identifier property pane, containing a real-time record of the entries. Methods for user and session authentication, permission or denial for data access, security and ownership management, highly selective data access and routing, Intelligent Object handling and storage are immediately provided.

[**0038**] An object root router component defines the origin of the object within the network, directs storage of the object within the database and reports the location of the object to the unique object identifier property pane.

[**0039**] An interactive content routing component defines where data content is located and where query-relevant content and/or results will be directed within the network for analysis or presentation and reports the location of the data content to the unique object identifier property pane.

[**0040**] A status management component provides methods for data status validation, logging, use-tracking, auditing, synchronization, rollback enabled by the command history and non-destructive vector processing, and other state management and alerting protocols. The status management component communicates with an external object state engine component to monitor data integrity and to record the command history according to G*P-compliant LIMS requirements (such as, for example, GLP, GCP, GMP, ISO-9001, CDER, CFIR) within the object state engine (OSE) property pane, where the information is updated and provided for real-time viewing. This information includes detailed activity logging, such as data acquisition state, calibration information, applied transformation or analysis processes, local and remote access attempts, access permission and denial, data integrity alerts, ranking status and regulatory validation states.

[**0041**] A raw data matrix property pane within the Intelligent Object provides an overview of the full raw data content subset including content attribute information, source location, data type and comments regarding data content referred to by the Intelligent Object, regardless of originating data type or structure.

[**0042**] A matrix structure descriptor component provides methods for data field mapping of heterogeneous raw data to govern access to individual data subsets (byte-level workspace assignment) and to enable direct vectorized access to individual data fields.

[**0043**] A meta-data index property pane within the Intelligent Object provides a viewer for automatically generated or user-defined index information and brief meta-descriptions ("data-about-data") such as, but not limited to, specific data functionality or relationships to other data or data inter-dependencies based upon multi-parametric clustering, queries or application of certain analytical tools or a combination thereof to the data. This pane utilizes a meta-data index interface to communicate with external processing engines to create an index of descriptive data information and to provide this meta-data to the object pane descriptor component, which integrates relevant pane information for access and presentation.

[**0044**] Additionally, the meta-data index is used to integrate results of clustering and/or other data analyses and provides sets of rules to optimize access and routing based upon dynamically established query relationship trees regarding specific data functionality and/or meta-data description. The meta-data index is also used to rank parameters such as data quality, validation state, significance, recency and accessibility to enable optimized access and routing based on data type, topic and content attributes to predefine analytical queries.

[**0045**] An object pane descriptor component compiles an overview of the Intelligent Object property pane characteristics to provide functional content and attribute definitions to access and route data content and applications. The object pane descriptor component exchanges the information with components and access interfaces to provide definitions required for dynamic addressing, functional linking and vectorized access and routing of data content and processing results.

[**0046**] The application translator link interacts with an external data and applications handler, such as the Intelligent Object Handler, and an object access manager component to provide a dynamic list and interactive overview of applications, application components and data resources. The application translation link enables interactive linking and integration of the applications, applications components and data resources, according to user requirements and available resources.

[**0047**] The object access manager determines relevant property panes and selectively directs their content for functional presentation and access within a given application or database environment. Additionally, the object access manager interacts with external object translation engines to detect, define and address required and/or available data sources and to direct access and routing requests for specific data content to linked applications and/or databases and to functionally integrate data content with a variety of applications. The object access manager also provides content-attribute based algorithms to enable applications integration and inter-application communication.

[**0048**] An object query interface routes results of Boolean comparisons and other algorithms applied to content attributes according to the Intelligent Object pane descriptor relationships. This component also passes aggregated results from object-to-object direct information interchange to an external result aggregation engine for further processing and relays significant query outcomes back to the object pane descriptor property pane for presentation to the user.

[**0049**] An object graph preview property pane is included as a limited resolution image/graphics viewer for quick graphical data review of Intelligent Objects, and additionally provides linking to detailed viewing as well as launching of content-specific analysis tools. The pane also includes an object graph preview processing component, which accounts for generation of such a limited resolution image/graphics ("thumbnail") from non-graphical raw data content and which passes the image back to the object graph preview property pane.

[**0050**] An optional text annotation property pane within the Intelligent Object provides a location for customized text annotations, referencing, definitions and integration of links to external textual resources. An optional text annotation interface links external components or applications such as text editors to allow for customization, formatting, reviewing and processing of the information through external editors and which allows to pass this information back to the text annotation property pane and to provide integrated support for text mining algorithms utilizing external distributed learning and/or knowledge extraction engines.

BRIEF DESCRIPTION OF THE DRAWINGS

[**0051**] **FIG. 1** depicts an embodiment of the unified user interface of an Intelligent Object showing its unique object identifier (UID) property pane.

[**0052**] **FIG. 2** depicts an embodiment of the user interface of an Intelligent Object showing its object state engine (OSE) property pane with the status management component report.

[**0053**] **FIG. 3** depicts an embodiment of the user interface of an Intelligent Object showing the object graph preview (OGP) property pane, comprising a limited resolution image/graphics viewer.

[**0054**] **FIG. 4** depicts an embodiment of the major elements, components, access interfaces and their relationships, showing the relationship of the Intelligent Object to an external Intelligent Object Handler (IOH), its components and access interfaces, the legacy domain of existing data content, applications, and devices, and an external Intelligent Object Pool (IOP).

[**0055**] **FIG. 5** depicts an embodiment of an exemplary hardware configuration for the Intelligent Object and its enabling architecture.

[**0056**] **FIG. 6** depicts an embodiment of Intelligent Objects comprised within an overall software platform architecture (Sentient IT Platform) for one exemplary and advantageous embodiment.

[**0057**] **FIG. 7** depicts an embodiment of Intelligent Objects providing a conceptual overview of multiple dimensions of (or "multidimensional") direct information interchange within and between Intelligent Objects.

DESCRIPTION OF EXEMPLARY EMBODIMENTS

[**0058**] The method described herein remedies problems and constraints identified in the Background by allowing interactive, Intelligent Object-based communication directly between functionally related attributes of heterogeneous data. This allows for comparison and extraction of all relevant content in a fast, unique and automated manner, within diverse network environments, without the need of upgrading or replacing current computer systems. The Intelligent Molecular Object technology provides a flexible global standard, which allows for seamless integration and real-time answers to complex, multidimensional and interdependent queries. Intelligent Molecular Object technology provides a framework for highly efficient scale-up and dynamically changing application needs in bioinformatics and the life sciences.

[**0059**] Methods are provided to define and describe a specific embodiment of architecture for Intelligent Object (IMO) **200** data structures. Intelligent Objects contain hierarchical, multi-layered property panes (Property Panes) **1000** for unified user presentation and functional interactivity, as well as components (Components) **1002** and access interfaces (Interfaces) **1004** to provide data status management, self-organizing data, and parallel data-to-data information interchange and processing, within local computing environments and/or over heterogeneous global computing networks.

[**0060**] This exemplary embodiment of the Intelligent Object (IMO) **200** provides interactive, secure, property-driven functional access to and integration of data content queried, presented and analyzed, utilizing a variety of raw data sources, applications and analytical components.

[**0061**] More particularly, aspects and embodiments of the invention provide (among others) an Intelligent Object, advantageously enabled in software, provides methods for:

[**0062**] (1) Data and user identification and status management, including:

[**0063**] Unified presentation and global unique identification of the Intelligent Object;

[**0064**] Identification, authentication and logging of users, sessions, and data content activity locally and/or over networks;

[**0065**] Dynamic routing of root object data, meta-data and data content locally and/or over networks;

[**0066**] Data status, data integrity and validation state alerting and management;

[**0067**] (2) Functional integration of, and access to, potentially diverse data over a variety of computing infrastructures, including:

[**0068**] Functional content and attribute definition for database and application access and routing;

[**0069**] Automated raw data matrix and matrix structure definition;

[**0070**] Automated translation of multiple data types and dimensions for unified processing and analysis;

[**0071**] Functional integration of multiple data types and dimensions for unified processing and analysis;

[**0072**] (3) Integration of multiple data types and dimensions for efficient and accurate multidimensional, parallel queries and analyses, including:

[**0073**] Meta-data indexing and query optimization;

[**0074**] Direct data-to-data information exchange;

[**0075**] Vectorized exchange of information subsets;

[**0076**] Data-enabled parallel processing;

[**0077**] Non-destructive cache-based processing;

[**0078**] Graphical data preview and detailed viewing;

[**0079**] Automated functional integration and launching of applications and activation of data related to Intelligent Object data content;

[**0080**] Automated assembly of applications and components for viewing and/or analysis relevant to specified data types and contents; and

[**0081**] Custom text annotation, linking and embedding of existing text.

[**0082**] These features and capabilities may be applied to diversified data content and dynamic applications needs in heterogeneous local and/or networked Life Sciences computing and applications environments, as well as to other applications and environments.

[0083] The invention provides structure, method, computer program and computer program product for an intelligent object. An Intelligent Object, advantageously enabled in computer program software code or instructions, provides methods for: data and user identification and status management; functional integration of, and access to, potentially diverse data over a variety of computing infrastructures; integration of multiple data types and dimensions for efficient and accurate multidimensional, parallel queries and analyses; for diversified data content and dynamic applications needs in heterogeneous local and/or networked computing and applications environments, especially in Life Sciences computing and applications environments.

[0084] Methods for creation of Intelligent Objects are provided which, upon user initiation, queries, data acquisition protocols or data import requests invoke the unique object identifier property pane through a unified functional presentation layer. The unique object identifier property pane assigns each new data object a globally unique identification upon creation and generates a minimum set of functional property panes within the object, which account for unified viewing and processing. Once the object's state recording is started, active identification for all connections to and activities on the Intelligent Object are listed within the unique object identifier property pane, containing a real-time record of the entries. Methods for user and session authentication, permission or denial for data access, security and ownership management, highly selective data access and routing, Intelligent Object handling and storage are immediately provided.

[0085] An object root router component defines the origin of the object within the network, directs storage of the object within the database and reports the location of the object to the unique object identifier property pane.

[0086] An interactive content routing component defines where data content is located and where query-relevant content and/or results will be directed within the network for analysis or presentation and reports the location of the data content to the unique object identifier property pane.

[0087] A status management component provides methods for data status validation, logging, use-tracking, auditing, synchronization, rollback enabled by the command history and non-destructive vector processing, and other state management and alerting protocols. The status management component communicates with an external object state engine component to monitor data integrity and to record the command history according to G*P-compliant LIMS requirements (such as, for example, GLP, GCP, GMP, ISO-9001, CDER, CFIR) within the object state engine (OSE) property pane, where the information is updated and provided for real-time viewing. This information includes detailed activity logging, such as data acquisition state, calibration information, applied transformation or analysis processes, local and remote access attempts, access permission and denial, data integrity alerts, ranking status and regulatory validation states.

[0088] A raw data matrix property pane within the Intelligent Object provides an overview of the full raw data content subset including content attribute information, source location, data type and comments regarding data content referred to by the Intelligent Object, regardless of originating data type or structure.

[0089] A matrix structure descriptor component provides methods for data field mapping of heterogeneous raw data to govern access to individual data subsets (byte-level workspace assignment) and to enable direct vectorized access to individual data fields.

[0090] A meta-data index property pane within the Intelligent Object provides a viewer for automatically generated or user-defined index information and brief meta-descriptions ("data-about-data") such as, but not limited to, specific data functionality or relationships to other data or data inter-dependencies based upon multi-parametric clustering, queries or application of certain analytical tools or a combination thereof to the data. This pane utilizes a meta-data index interface to communicate with external processing engines to create an index of descriptive data information and to provide this meta-data to the object pane descriptor component, which integrates relevant pane information for access and presentation.

[0091] Additionally, the meta-data index is used to integrate results of clustering and/or other data analyses and provides sets of rules to optimize access and routing based upon dynamically established query relationship trees regarding specific data functionality and/or meta-data description. The meta-data index is also used to rank parameters such as data quality, validation state, significance, recency and accessibility to enable optimized access and routing based on data type, topic and content attributes to predefine analytical queries.

[0092] An object pane descriptor component compiles an overview of the Intelligent Object property pane characteristics to provide functional content and attribute definitions to access and route data content and applications. The object pane descriptor component exchanges the information with components and access interfaces to provide definitions required for dynamic addressing, functional linking and vectorized access and routing of data content and processing results.

[0093] The application translator link interacts with an external data and applications handler, such as the Intelligent Object Handler, and an object access manager component to provide a dynamic list and interactive overview of applications, application components and data resources. The application translation link enables interactive linking and integration of the applications, applications components and data resources, according to user requirements and available resources.

[0094] The object access manager determines relevant property panes and selectively directs their content for functional presentation and access within a given application or database environment. Additionally, the object access manager interacts with external object translation engines to detect, define and address required and/or available data sources and to direct access and routing requests for specific data content to linked applications and/or databases and to functionally integrate data content with a variety of applications. The object access manager also provides content-attribute based algorithms to enable applications integration and inter-application communication.

[0095] An object query interface routes results of Boolean comparisons and other algorithms applied to content attributes according to its object pane descriptor relationships. This component also passes aggregated results from object-to-object direct information interchange to an external result aggregation engine for further processing and relays significant query outcomes back to the object pane descriptor property pane for presentation to the user.

[0096] An object graph preview property pane is included as limited resolution image/graphics viewer for quick graphical data review of Intelligent Objects, and additionally provides linking to detailed viewing as well as launching of content-specific analysis tools. The pane also includes an object graph preview processing component, which accounts for generation of such a limited resolution image/graphics ("thumbnail") from non-graphical raw data content and which passes the image back to the object graph preview property pane.

[0097] An optional text annotation property pane within the Intelligent Object provides a location for customized text annotations, referencing, definitions and integration of links to external textual resources. An optional text annotation Interface, which links external components or applications such as text editors to allow for customization, formatting, reviewing and processing of the information through external editors and which allows to pass this information back to the text annotation property pane and to provide integrated support for text mining algorithms utilizing external distributed learning and/or knowledge extraction engines.

[0098] For reasons of explanation, these methods, components and processes will be described in a fashion that does not represent the entity of simultaneous and/or interactive actions as they occur. However, it should be noted that the system herein described is composed of bi-directionally interactive components and interfaces which perform certain tasks simultaneously, or in a rapidly alternating fashion.

[0099] Examples of enabling code are provided to define and describe a single exemplary embodiment, which utilizes Microsoft C++ as the exemplary programming language.

[0100] Additionally, software development tools not limited to Visual C++, Microsoft Foundation Classes (MFC), DIB image transformations and matrix-based graphical content generation were utilized to enable this specific embodiment. The overall architecture, its application across varied domains, its processing engines and its access interfaces are in no way limited to the utilization of Microsoft C++ or the Windows 32-bit operating system environment. It is readily apparent to anyone skilled in the art that other enabling software codes or enabling techniques may also be used, including for example Java, XML and other markup languages, and/or other similar techniques.

[0101] The Intelligent Object may be compiled to run on multiple platforms, including, but not limited to, UNIX, Linux, Macintosh OS 9 and 10, or any Window 32-bit operating systems. The following hardware specifications are provided to define and describe the requirements for a specific exemplary embodiment, implemented for a 32-bit Microsoft Windows environment.

[0102] The inventive system, architecture, method, and computer program and computer program product of the Intelligent Object as well as other core elements and modules described herein and in the related applications identified herein, may be used in a variety of computing and network or connectivity environments as are known in the art and advantageously are hardware and operating system agnostic. For example, the invention may be practiced with the great majority of contemporary personal computers, workstations, mainframes, as well as notebook and other portable computing devices and all manner of information appliances.

[0103] It is clear to anyone skilled in the art that these requirements are provided by way of instruction regarding a specific embodiment of the technology, and that the implementation of the Intelligent Object is not limited only to the particular embodiments described.

[0104] Aspects and features of the invention are now demonstrated and described relative to the interaction a scientist or other investigator or user would have with the inventive system, method, and computer program or computer program interface. In this demonstration example, data from 2-Dimensional Gel Electrophoresis (2DE) typically exhibit an intrinsic complexity due to the reproducibility challenges inherent in this multi-step experimental technique. Each of such gel comprises over 2000 individual peptide spots that relate in its entity to a defined stage in the cell metabolism. Such image data were used in a global query to obtain characteristics of significant protein expression in human liver cells at different disease states. Only larger peptides with isoelectric points (pI's) between 5.0-7.0 and within a size range of >96000 Dalton (DA) were of therapeutic interest and only validated experiments were included.

[0105] With reference to **FIG. 1**, there is shown a representation of the user interface implementation of an Intelligent Object (IMO) showing its unique object identifier pane (UID). It contains Intelligent Object creation data; the location of the Intelligent Object on the network; information routing information; user data, session and connection verification and security settings such as specific encryption level, password protected or locally restricted data access.

[0106] The depiction in **FIG. 2** is a representation of the user interface implementation of an Intelligent Object (IMO) showing its object state engine (OSE) property pane and its comprise status management component. In the displayed example, the experimental state is validated and the raw data integrity verified. Means for a quick review of the Intelligent Object history are provided including significant events since its creation such as calibration, analysis and annotation as well as alerts such as access violations.

[0107] The illustration in **FIG. 3** depicts an intelligent object graph preview (OGP) property pane example for 2-Dimensional Gel Electrophoresis protein expression data. The low-resolution image viewer depicts an overview of the entire gel. The tree-styled content attribute in the right side of the upper part in the object pane describes the analytical techniques used in the experiment. A button "View And Analyze" launches appropriate application modules and/or analytical tools for non-destructive data analysis. The lower part of the pane contains descriptive comments about the content inserted by the creator of the Intelligent Object. With reference to **FIG. 4**, one embodiment of the inventive intelligent object, also or alternatively referred to as the intelligent molecular object (IMO), is described. Other embodiments may include selected elements from the **FIG. 4** embodiment, and/or have additional added elements. The depiction in **FIG. 4** represents elements of the invention and their relationships, showing the relationship of the Intelligent Object to an external Intelligent Object Handler (IOH), its components and access interfaces, the legacy domain of existing data content, applications, and devices, and an external Intelligent Object Pool (IOP). In the figure, unbroken lines ending with arrows on each end represent bi-directional communication between exemplary property panes, components and access interfaces; dashed lines ending with arrows on each end represent bi-directional communication between optional property panes, components and access interfaces; and, crossed lines do not represent connections.

[**0108**] In order that there be some appreciation for the manner in which computer hardware, local and/or global networks, public and private data resources and databases interoperate, **FIG. 5**, **FIG. 6**, and **FIG. 7** represents exemplary hardware and architecture configuration for the Intelligent Object and its enabling architecture and interactive content routing features.

[**0109**] With reference to **FIG. 5**, all major elements within the diagram below may be bi-directionally connected over a variety of network protocols. The minimum hardware requirement is defined by a single machine. In an exemplary embodiment, as below, two laptop computers are connected in a client/server configuration to a workstation to one another in a peer-to-peer configuration, and via the workstation directly to a laboratory instrument, such as a gene sequencer or gel electrophoresis machine. In this figure, dotted bi-directional lines 248 represent options for "any-to-any" connectivity enabled via use of Intelligent Objects as central accessing and routing components. Any-to-any options include but are not limited to LAN, WAN, peer-to-peer (e.g. data, applications, memory and processor sharing between two or more laptops, workstations, etc.), server-server, Portal, ASP and other unified, distributed, parallel and grid network options. Connectivity protocols include and are not limited to PPP, http, TCP/IP, and ftp over multiple platforms. With reference to **FIG. 6**, there is illustrated a representation of an embodiment of Intelligent Objects comprised within an exemplary software platform architecture (Sentient IT Platform). This embodiment depicts the Intelligent Object (IMO) comprised as a core element group, along with components (Processing Engines) and access interfaces required for the Sentient IT Platform.

[**0110**] **FIG. 7** depicts an embodiment of Intelligent Objects providing a conceptual overview of interactive content routing for multiple dimensions of (or "multidimensional") direct information interchange within and between Intelligent Objects. In the embodiment of **FIG. 7**, each property pane provides an overview of certain properties of the comprised data and its relationships. For example, property panes describe Intelligent Object ownership and activity history, but also complex, multiple relationships to other data and applications. Vectorized data content information and results of data content comparison and analysis, data annotation, text references, validation status and the like may be flexibly interconnected in a functional manner via these panes and their related components and access interfaces. User viewing and interactivity to define or refine (without writing to the data content) Intelligent Object property pane content presentation and relationship connectivity for new queries, customization and the like takes place through property panes presented at the unified presentation layer.

[**0111**] The following exemplary embodiment is defined by a core set of processing components and access interfaces described in detail below. Alternative embodiments may or may not have corresponding or additional processing components, access interfaces and property panes with unique, functionality-driven properties. In this exemplary embodiment, the set of property panes **1000**, processing components **1002** and their corresponding access interfaces **1004** are defined as follows.

[**0112**] A unique object identifier (UID) **1006** property pane enables unified user viewing of object creation information, e-signature authentication, access privilege information and common security elements related to each data object and to corresponding data content. This information comprises a 128-bit, alphanumeric, globally unique intelligent object identification string, unique user identification, unique computer identification, unique session identification, login time, user network address, and date and time record of the start of the last connected session (See **FIG. 2**). The Intelligent Object's (IMO) **200** creator/owner, routing information and content attributes as well as information on currently connected users, current session information, access permissions, authentication, routing and are reported to and presented via this pane and are updated dynamically.

[**0113**] An interactive content router (ICR) **1008** component bi-directionally communicates with the unique object identifier and also interacts with the object pane descriptor (OPD) **1026** to define where data content is located, where and how it will be accessed and directed within the network, and which distributes data content according to a given queries, reporting the specific content origination and routing information to the unique object identifier (UID) **1006** property pane.

[**0114**] An object root router (ORR) **1010** component bi-directionally communicates with the unique object identifier interface (UIDi) **1012** and interactive content router (ICR) **1008** to define the origin of the Intelligent Object (IMO) **200** within the network, addresses object queries and reports this information to the unique object identifier (UID) **1006** property pane. The object root router (ORR) **1010** enables secure local and/or remote identification of and interactive access to the Intelligent Object, utilizing and reporting access and routing information comprising the unique object identifier, data content and data object ranking and ownership information, object root addressing and routing information, activity logging and reporting of all connected users.

[**0115**] A unique object identifier access interface (UIDi) **1012** gates direct access and routing to all or selected areas of the data content, according to authentication and permission and/or denial based on unique data, user and session authentication, and Intelligent Object (IMO) **200** security and content routing permissions.

[**0116**] An example of enabling code for the implementation of the unique object identifier (UID) **1006** property pane is shown in the following.

[**0117**] Various selected aspects and features of the invention are now described relative to six examples. It will be appreciated that these examples are not intended to cover, address, or describe all of the inventive features. Example 1 shows a specific embodiment of enabling code, providing instructions utilized for the exemplary embodiment of the unique object identifier (UID) processing component.

Example 2 shows a specific embodiment of enabling code, providing instructions utilized for the exemplary embodiment of the interactive content router (ICR) processing component. Example 3 shows a specific embodiment of enabling code, providing instructions utilized for the exemplary embodiment of the object state engine (OSE) property pane and its status management component. Example 4 shows a specific embodiment of enabling code, providing instructions utilized for the exemplary embodiment for opening of multiple object data with similar content within a defined Intelligent Object directory. Example 5 shows a specific embodiment of enabling code, providing instructions utilized for the exemplary embodiment of the object graph preview (OGP) property pane and its module launch component. Example 6 shows a specific embodiment of enabling code, providing instructions utilized for the exemplary embodiment of the raw data matrix (RDM) property pane and its data link insertion component.

EXAMPLE 1

[0118] Using message mapping, the IMO UID pane is initialized to receive the following CString data via DoDataExchange command: a unique object UID as a 128 bit global identifier string; an entry on the last connection to the object, including login time, date, network, computer address and user identifier; a counter for concurrent connections to the object and a list containing information on each connection; and the object creator string containing the unique user global identifier.

[0119] This information is then displayed within the UID pane and refreshed on update:

```
    DWORD flags = m_listConnection.GetExtendedStyle();
    m_listConnection.SetExtendedStyle(flags | LVS_EX_FULLROWSELECT| LVS_EX_GRIDLINES);
    m_listConnection.InsertColumn(0, _T("User"), LVCFMT_LEFT);
    m_listConnection.InsertColumn(1, _T("Computer Name"), LVCFMT_LEFT);
    m_listConnection.InsertColumn(2, _T("Login Time"), LVCFMT_LEFT);
    m_listConnection.InsertColumn(3, _T("IP Address"), LVCFMT_LEFT);
    m_listConnection.InsertColumn(4, _T("Session ID"), LVCFMT_LEFT);
    m_listConnection.SetColumnWidth(0, LVSCW_AUTOSIZE_USEHEADER);
    m_listConnection.SetColumnWidth(1, LVSCW_AUTOSIZE_USEHEADER);
    m_listConnection.SetColumnWidth(2, LVSCW_AUTOSIZE_USEHEADER);
    m_listConnection.SetColumnWidth(3, LVSCW_AUTOSIZE_USEHEADER);
    m_listConnection.SetColumnWidth(4, LVSCW_AUTOSIZE_USEHEADER);
    UpdateData(FALSE);
        return TRUE;       // return TRUE unless you set the focus to a control
                           // EXCEPTION: OCX Property Pages should return FALSE
(End Example 1)
```

[0120] Object root routing and interactive content routing are permitted or denied according to unique user, session, data object and data content identification enabled by the object root router (ORR) **1010** and interactive content router (ICR) **1008** components to provide finely grained access and routing control, definable as narrowly as to single bytes of data content. The interactive content routing (ICR) **1008** component also assures via the data content access and routing protocols that analytical and/or write requests are not allowed to the created data file content, but rather are implemented utilizing cache-based non-destructive overlay processing methods enabled via an external Intelligent Object Handler (IOH) **202**.

[0121] Additional methods of data encryption may be provided in optional embodiments, to protect the Intelligent Object (IMO) **200** and its contents from unauthorized access. The methods of data encryption are well known in the art and examples of comprised data encryption methods include RSA Data Security software solutions of Redwood City, Calif., USA.

[0122] An example of enabling code for the interactive content routing component (ICR) **1008** is shown in the following.

EXAMPLE 2

[0123] Interactive content routing involves calls to the document manager to filter information based on its content attribute and to activate loading of certain raw data structures and/or applications selectively without user interaction.

```
BOOL  CDocManagerEx::DoPromptFileName(CString& fileName, UINT nIDSTitle, DWORD lFlags,
BOOL bOpenFileDialog, CDocTemplate* pTemplate)
{
    CFileExDlg dlgFile(bOpenFileDialog); // dialog with a preview
    CString title;
    VERIFY(title.LoadString(nIDSTitle));
    dlgFile.m_ofn.Flags |= lFlags;
    CString strFilter;
    CString strDefault;
```

-continued

```
    if (pTemplate != NULL)
    {
        ASSERT_VALID(pTemplate);
        _AfxAppendFilterSuffix(strFilter, dlgFile.m_ofn, pTemplate, &strDefault);
    }
    else
    {
        // do for all doc template
    POSITION pos = m_templateList.GetHeadPosition();
    BOOL bFirst = TRUE;
    while (pos != NULL)
    {
        CDocTemplate* pTemplate = (CDocTemplate*)m_templateList.GetNext(pos);
        _AfxAppendFilterSuffix(strFilter, dlgFile.m_ofn, pTemplate,
            bFirst ? &strDefault : NULL);
        bFirst = FALSE;
    }
}
```

[0124] A function is provided for content-based load and preview of files within the IMO content panes. The content descriptor is located in "section 4" of the structure.

```
// open file and look inside
CFile f;
VERIFY(f.Open(ff.GetFilePath(), CFile::modeRead));
if(CMainDoc::IsFileValid(&f) == FALSE)
    continue;
// don't support automatic opening for version 1
if(CMainDoc::GetFileVersion(&f) < 2)
    continue;
      TRY
            {
    // Descriptor is at section 4
    f.Seek(3 + ((4 - 1) * sizeof(DWORD)), CFile::begin);
    DWORD position = 0;
    f.Read(&position, sizeof(position));
    //seek to the right position and start reading
    f.Seek(position, CFile::begin);
      // section version
      BYTE version = 0;
      f.Read(&version, sizeof(version));
      ASSERT(version == 1);
        CObjectPaneDescriptorArray opd;
        CArchive ar(&f, CArchive::load |
        CArchive::bNoFlushOnDelete);
        ar.m_pDocument = NULL;
        ar.m_bForceFlat = FALSE;
        opd.Serialize(ar);
        ar.Close();
        if(opd.GetSize() <= 0)
          continue;
        if(opd[0].shTechnique != m_nCurrContentAttrib)
          continue;
      }
          CATCH(CException, e)
            {
        TRACE("Could not read this file: %s\n",ff.GetFilePath());
        continue;
          }
            END_CATCH
        TRACE("Other file to open: %s\n",ff.GetFilePath());
        m_arrayFileName.Add(ff.GetFilePath());
        count++;
      }
    }
    return FALSE;
}
```

[0125] A preview thumbnail of the data is contained in "section 5" of the structure.

```
    // thumbnail is at section 5
    f.Seek(3 + ((5 - 1) * sizeof(DWORD)), CFile::begin);
    DWORD position = 0;
    f.Read(&position, sizeof(position));
    // seek to the right position and start reading
    f.Seek(position, CFile::begin);
    // section version
    BYTE version = 0;
    f.Read(&version, sizeof(version));
    ASSERT(version == 1);
    // thumbnail image
    BYTE raw_data[240 * 240 * 3]= {0};
    f.Read(raw_data, sizeof(raw_data));
    L_CreateBitmap(&m_bitmapThumbnail, TYPE_CONV,
                240, 240, 24,
                ORDER_BGR, NULL, TOP_LEFT);
    int width = BITMAPWIDTH(&m_bitmapThumbnail);
    int height = BITMAPHEIGHT(&m_bitmapThumbnail);
    RGBTRIPLE* rgb = (RGBTRIPLE*)raw_data;
    for(int row = 0; row < height; row++)
        for(int col = 0; col < width; col++)
        {
            COLORREF cr = RGB(rgb[(row * height) + col].rgbtRed,
                    rgb[(row * height) + col].rgbtGreen,
                    rgb[(row * height) + col].rgbtBlue);
            L_PutPixelColor(&m_bitmapThumbnail, row, col, cr);
        }
.....
(End Example 2)
```

[0126] An object state engine (OSE) **1014** (also see **FIG. 2**) property pane is enabled by the status management component (SMC) **208** and an object state list (Table 1) to provide real-time user viewing and integrity verification of object state and object command history information including access attempts, access permission and denial, detailed activity logging, data integrity alerts, ranking status and regulatory validation states. Table 1 is an exemplary representation of the object state list, showing the coded table references utilized to minimize overhead for regulatory-compliant state synchronization and logging.

[0127] A status management component (SMC) **208** communicates with an external object state engine component (OSE) **212** to monitor data integrity and command history according to industry specific, regulatory compliant parameters. The status management component (OSE) **208** provides information including data acquisition state (instrument parameters, acquisition completion, etc.), calibration information, applied transformation or analysis processes and validation state (object state list—Table 1). Information and contained herein is used for data integrity protection, regulatory validation, auditing and logging, rollback.

[**0128**] An object query interface (OQI) **1018** (OQI) **1018** receives query requests from the external object state engine (OSE) **212** and initiates Intelligent Object (IMO) **200** meta-data analysis and provides access interfaces between Intelligent Object (IMO) **200** processing components and external processing components and access interfaces, according to the correspondence of meta-data indices and detected content attributes to a given query. These components and interfaces may include but are not limited to a report generation interface (RGI) **220**, a result aggregation engine (RAE) **204**, validated and/or functionally ranked analytical processing components and the object query interfaces (OQI) **1018** of related data.

[**0129**] Additionally, the object query interface (OQI) **1018** enables object-to-object direct information interchange by directing interactive content routing of specified data vectors directly between Intelligent Objects (IMO) **200** and relaying results of direct object-to-object communication and comparison to an external result aggregation engine (RAE) **204** component and external report generation interface (RGI) **220**.

[**0130**] Examples of enabling code for the implementation of the object state engine (OSE) **1014** property pane and status management component (SMC) **208** are shown in the following.

EXAMPLE 3

[**0131**] The object state engine property pane (OSE) **1014** handles state-related information for access and validation. CStrings for state history record counter, alert counter and current state records are created.

```
void CPageOseDlg::DoDataExchange(CDataExchange* pDX)
{
    CString count_str = _T("N/A");
    CString alert_str = _T("N/A");
    CString alert_count_str = _T("N/A");
    CString curr_state_str = _T("N/A");
    if(pDX->m_bSaveAndValidate == FALSE)
    {
        CDialogView* view = (CDialogView*)GetParent();
        CMainDoc* doc   = (CMainDoc*)view->GetDocument();
        ASSERT_VALID(view);
        ASSERT_VALID(doc);
        if(::IsWindow(m_listObjectHistoy.m_hWnd))
        {
            // do the list
            m_listObjectHistory.DeleteAllItems();
            int alert = 0;
            int last_alert = 0;
            int last_state = 0;
            time_t last_alert_time = 0;
            time_t last_state_time = 0;
            CString text = _T("");
            CTime time = 0;
            UUIDWSTR uuid_str = {0};
            BYTE flag   = 0;
            CString temp = _T("");
            int txt_width = 0;
            int col_width = 0;
            int count = doc->m_arrayStateHistory.GetSize();
            for(int i = 0; i < count; i++)
            {
                GetLutObjState(doc->m_arrayStateHistoty[i].wState, temp, flag);
                ... ...
                      ASSERT(((flag & 8) && (flag & 16)) != TRUE);
                m_listObjectHistory.SetItemData(i, flag);
                if((flag & 8) || (flag & 16))
                {
                    alert++;
                    last_alert = doc->m_arrayStateHistory[i].wState;
                    last_alert_time = doc->m_arrayStateHistory[i].timeDate;
                }
            // set the list count and alerts
            count_str.Format(_T("%d"), count);
            alert_count_str.Format(_T("%d"), alert);
            // set the text for last alert
            if(alert > 0)
            {
                GetLutObjState(last_alert, temp, flag);
                time = last_alert_time;
```

-continued

```
            alert__str.Format(__T("%s on %s"), temp, time.Format(__T("%x")));
        }
        else
            alert__str = __T("None");
        // set text for the state the object is in
        if(count > 0)
        {
        GetLutObjStateParent(last__state, temp, flag);
        time = last__state__time;
        curr__state__str.Format(__T("%s on %s"), temp, time.Format(__T("%x")));
    }
    else
        curr__state__str = __T("None");
    }
}
```

[0132] The status management component (SMC) **208** uses a hierarchical tree-style lookup table (LUT) for state assignment.

```
// create the tree
BYTE flag = 0;
CString str = __T("");
HTREEITEM parent = NULL;
for(int i = 0; i < 1000; i++)
{
    GetLutObjState(i, str, flag);
    if(!str.IsEmpty())
    {
        ASSERT(((flag & 8) && (flag & 16)) != TRUE);
        if(flag & 1)
        {
            parent = m__treeObjectSate.InsertItem(str);
            m__treeObjectSate.SetItemData(parent, flag);
            continue;
        }
        if(flag & 2)
        {
            CString fmt = __T("");
            fmt.Format(__T("(%.3d) - %s"), i, str);
                HTREEITEM item = m__treeObjectSate.InsertItem(fmt, parent);
                m__treeObjectSate.SetItemData(item, flag);
                continue;
            }
            ASSERT(FALSE);
        }
    }
            return TRUE; // return TRUE unless you set the focus to a control
                    // EXCEPTION: OCX Property Pages should return FALSE
}
(End Example 3)
```

[0133] A meta-data index property pane (MDX) **1022** enables unified user viewing and customization of object meta-data definitions for optimized query processing. Meta-data information includes specific data functionality, content attributes and relationships to other data derived from a variety of statistical comparisons such as clustering and self-organizing maps, as well as from query histories and other user-based information, to predefine searching and analysis of Intelligent Objects (IMO) **200**.

[0134] A meta-data index access interface (MDXi) **1022** allows for fast access to the Intelligent Object (IMO) **200** based on data content, functionality and description. The meta-data index interface provides direct linking to indexed and ranked "data-about-data" information to enable optimized access and routing.

[0135] An object pane descriptor property pane (OPD) **1024** provides an interactive overview of the Intelligent Object's (IMO) **200** property panes, data content, location, structure and functional relationships of linked applications and linked databases provided by an application/database definition router interface.

[0136] An object pane descriptor component (OPDc) **1026** compiles an overview of Intelligent Object (IMO) **200** property pane (Property Panes) **1000** characteristics to provide functional content and attribute definitions to access and route data content and applications. The object pane descriptor component (OPDc) **1026** exchanges this information with components and access interfaces including but not limited to the interactive content router component (ICR) **1008**, the status management component (SMC) **208**, the object query interface (OQI) **1018**, the meta-data access interface (MDXi) **1022**, an object access manager component (OAM) **1036**, an application/database definition router (ADDR) **1028** and an application translation linking component (ATL) **1034** to provide definitions required for dynamic addressing, functional linking and vectorized access and routing of data content and processing results.

[**0137**] An application/database definition router (ADDR) **1028** enables the detection, definition and addressing of required and/or available data sources and directs access and routing requests to specific data content, linked applications and linked databases. The application/database definition router (ADDR) **1028** interactively provides this information as required for a given context and query, interacting with the application translation linking component (ATL) **1034**, the interactive content routing component (ICR) **1008** and the object query interface (OQI) **1018** to call proper analysis tools based on the Intelligent Object (IMO) **200** property pane information (Property Panes) **1000** available.

[**0138**] An object graph preview property pane (OGP) **1030** within the Intelligent Object (IMO) **200** includes linking to detailed viewing and analysis tools (See **FIG. 3**), as well as a limited resolution image/graphics ("thumbnail") viewer for quick graphical review of the raw data corresponding to the Intelligent Object (IMO) **200**.

[**0139**] An object graph preview interface (OGPi) **1032** routes content attribute information, comments, and a limited resolution graphic view of any selected raw data file to the object graph preview property pane (OGP) **1030**. Additionally, specific content attributes and comments may be viewed, as well as subsets of data with corresponding content attributes, upon user request.

[**0140**] Examples of enabling code for the implementation of the object graph preview property pane (OGP) **1030** and module launching methods are shown in the following.

EXAMPLE 4

[**0141**] The object graph preview property pane (OGP) **1030** provides a thumbnail graphical preview of the data content within the object based on information in the object pane descriptor (OPDC) **1026** regarding technique and content attribute.

```
if(doc->m_arrayObjectPaneDescriptor.GetSize() > 0)
{
    BOOL create = FALSE;
    if(m_strContentAttrib.IsEmpty() == FALSE)
    {
            int index1 =
            doc->m_arrayObjectPaneDescriptor[0].shTechnique;
            int index2 = atoi(m_strContentAttrib.Mid(1,3));
            if(index1 != index2)
                create = TRUE;
    }
    else
            create = TRUE;
        if(create == TRUE)
            CreateContentAttribString();
    }
}
```

[**0142**] The content is referred to according to a hierarchical tree-style attribute lookup table (LUT) similar to the one used for object state management. Bit-flags are provided to define the depth of the tree branches.

```
    // look up info in the tree
    GetLutContentAttrib(doc->m_arrayObjectPaneDescriptor[0].shTechnique, str, flag);
    m_strContentAttrib.Format(_T("(%.3d)  \\%s"), doc->m_arrayObjectPaneDescriptor[0].shTechnique,
str);
    parent = tree.GetParentItem(item);
    while(parent)
    {
        m_strContentAttrib.Insert(6,_T("\\") + tree.GetItemText(parent));
        parent = tree.GetParentItem(parent);
    }
}
```

[**0143**] The object graph preview access interface (OGPi) **1032** contains instructions for module launching, used for dynamic, content-attribute based launch of appropriate analysis tools. Applications are called according to their index grouping via a dynamically generated command line string.

```
void CPageOgpDlg::OnAnalyze()
{
    CDialogView* view = (CDialogView*)GetParent();
    ASSERT_VALID(view);
    CMainDoc* doc = (CMainDoc*)view->GetDocument();
    ASSERT_VALID(doc);
    // make we have the right analysis tools before we continue
```

-continued

```
    CArray<int, int>range;
    int doc__attrib = doc->m__arrayObjectPaneDescriptor[0].shTechnique;
    ASSERT(doc__attrib != 0);
    ASSERT(range.GetSize() == 0);
    if((doc__attrib >= 210) && (doc__attrib <= 269))
    {
        for(int i = 210; i <= 269; i++)
            range.Add(i);
    }
    else if((doc__attrib >= 280) && (doc__attrib <= 314))
    {
        for(int i = 280; i <= 314; i++)
            range.Add(i);
    }
.....
    // analysis not supported
    if(range.GetSize() == 0)
    {
        AfxMessageBox(IDS__ERROR__WRONGCONTENTATTRIB);
        return;
    }
    // get the executable file path
    TCHAR file__path[__MAX__PATH] = {0};
    VERIFY(GetModuleFileName(AfxGetApp()->m__hInstance, file__path, __MAX__PATH));
    TCHAR drive[__MAX__DRIVE] = {NULL};
    TCHAR dir[__MAX__DIR] = {NULL};
    TCHAR title[__MAX__FNAME] = {NULL};
    TCHAR ext[__MAX__EXT] = {NULL};
    __tsplitpath(file__path, drive, dir, title, ext);
#if defined(__DEBUG)
    CString fn = __T("");
    fn += title;
    fn += ext;
    ASSERT(fn.CompareNoCase("Platform.exe") == 0);
#endif
    CString process__path = __T("");
    process__path += drive;
    process__path += dir;
    process__path += __T("Analyze.exe");
....
    // convert command line string
    param.TrimRight();
    TCHAR* icr = new TCHAR[param.GetLength() + 1];
    memset(icr, NULL, sizeof(TCHAR) * (param.GetLength() + 1));
    __tcscpy(icr, param);
    // invoke process
    PROCESS__INFORMATION pi = {0};
    STARTUPINFO si = {0};
    si.cb = sizeof(si);
    CreateProcess(process__path, icr, NULL, NULL, FALSE, CREATE__NEW__CONSOLE,
        NULL, NULL, &si, &pi);
    //release buffer
    delete[] icr;
}
(End Example 4)
```

[0144] Additionally, the object graph preview interface (OGPi) **1032** detects data objects with corresponding content attributes to present limited resolution graphic previews of related data objects upon user request. Further, the object graph preview interface (OGPi) **1032** provides linking to analytical tools for detailed viewing.

[0145] Examples of enabling code, providing instructions utilized for the exemplary embodiment, for opening of multiple object data with similar content within a defined Intelligent Object (IMO) **200** directory are shown in the following.

EXAMPLE 5

[0146] A dynamic loading based on content is provided which allows to allocate and open all IMO data of similar content attributes simultaneously.

```
IMPLEMENT__DYNAMIC(CFileExDlg, CFileDialog)
BEGIN__MESSAGE__MAP(CFileExDlg, CFileDialog)
```

-continued

```
        //{{AFX_MSG_MAP(CFileExDlg)
    ON_WM_PAINT()
        ON_BN_CLICKED(IDC_OPENALLCURRENTATTRIB, OnOpenAllSameAttrib)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
CFileExDlg::CFileExDlg(BOOL     bOpenFileDialog,     CWnd*     /*     pParentWnd */) :
CFileDialog(bOpenFileDialog)
{
    m_ofn.Flags = m_ofn.Flags | OFN_FILEMUSTEXIST | OFN_PATHMUSTEXIST | OFN_EXPLORER |
OFN_ENABLETEMPLATE;
m_ofn.Flags &=~OFN_ENABLESIZING;
m_ofn.lpTemplateName = MAKEINTRESOURCE(CFileExDlg::IDD);
//{{AFX_DATA_INIT(CFileExDlg)
        m_strComments =_T("");
        m_strContentAttrib =_T("");
        m_bOpenAllSameAttrib = FALSE;
        m_bOpenAll = FALSE;
        //}}AFX_DATA_INIT
    m_nCurrContentAttrib = -1;
    m_arrayFileName.RemoveAll();
    memset(&m_bitmapThumbnail, NULL, sizeof(m_bitmapThumbnail));
}
void CFileExDlg::DoDataExchange(CDataExchange* pDX)
{
        CFileDialog::DoDataExchange(pDX);
    //{{AFX_DATA_MAP(CFileExDlg)
        DDX_Control(pDX, IDC_THUMBNAILFRAME, m_wndThumbnail);
        DDX_Text(pDX, IDC_COMMENTS, m_strComments);
        DDX_Text(pDX, IDC_CONTENTATTRIB, m_strContentAttrib);
        DDX_Check(pDX, IDC_OPENALLCURRENTATTRIB, m_bOpenAllSameAttrib);
        //}}AFX_DATA_MAP
}
BOOL CFileExDlg::OnInitDialog()
{
        CFileDialog::OnInitDialog();
    ClearPreview();
        return TRUE; // return TRUE unless you set the focus to a control
        // EXCEPTION: OCX Property Pages should return FALSE
}
....
```

[0147] Preview the selected IMO data's graphical thumbnails:

```
//find out the bitmap dimensions and set the frame size to the
bitmap size
m_wndThumbnail.SetWindowPos(NULL, 0, 0,
    BITMAPWIDTH(&m_bitmapThumbnail),
    BITMAPHEIGHT(&m_bitmapThumbnail),
    SWP_NOMOVE \ SWP_NOZORDER);
//paint the thumbnail
CClientDC wnd_dc(&m_wndThumbnail);
HPALETTE hp = L_CreatePaintPalette(wnd_dc.GetSafeHdc(),
&m_bitmapThumbnail);
CPalette *old_pal = wnd_dc.SelectPalette(CPalette::FromHandle(hp),
FALSE);
wnd_dc.RealizePalette();
L_PaintDC(wnd_dc.GetSafeHdc(), &m_bitmapThumbnail,
    NULL,
    NULL,
    CRect(0, 0, BITMAPWIDTH(&m_bitmapThumbnail),
    BITMAPHEIGHT(&m_bitmapThumbnail)),
    NULL,
    SRCCOPY);
wnd_dc.SelectPalette(old_pal, FALSE);
DeleteObject(hp);
}
(End Example 5)
```

[0148] An application translation link property pane (ATL) **1034** is linked to the object access manager component (OAM) **1036** to provide an interactive, unified user overview of applications components and relationships required and/or available, depending on the data content, functional data relationships, analytical requirements, translation requirements and specific user queries. This property pane is linked to an external component for automated applications assembly to enable quick and seamless application integration and to provide the functionality for automated and dynamic development of new applications.

[0149] An object access manager processing component (OAM) **1036** provides algorithms to allow for applications integration and inter-application communication, depending on data content attributes (Table 2—Content Attribute List). Table 2 is a representation of an embodiment or representation of the content attribute list, showing the coded table references utilized to minimize overhead for applications linking and object-to-object direct information interchange. Additionally, the object access manager (OAM) **1036** is linked to an external object translation engine (OTE) **214** component and application translation interface (ATI) **216** which optimize translation of the Intelligent Object (IMO) **200** and functionally integrate data content with a variety of applications.

[0150] A raw data matrix property pane (RDM) **1042** containing path and vectorized access information to diverse types of data content is provided. The raw data matrix (RDM) **1042** property pane provides an interactive interface to, in this exemplary embodiment, an external Intelligent Object Handler (IOH) **202** and a data link insertion (DLI) **1044** component to provide a unified user overview of the full information subset of the data content structure and format, including data content attribute information, data source location, data source type and comments regarding data content.

[0151] Through the data link insertion component (DLI) **1044**, data are distinguished and defined within this pane by their type membership (e.g., file, database record, application, life data acquisition, . . .), their transport and routing protocol and communication mechanism to the raw data (e.g., via PPP over TCP/IP, P2P, CIS, SIS; and/or via http, ftp, or other protocols via automatic multi-transport protocol detection) and their content attribute.

[0152] A matrix structure descriptor (MSD) **1046** component contains the full information subset, regarding the formatting and/or structure of the data content. This information is provided by an external object translation engine (OTE) **214** and applications translation interface (ATI) **216** and is relayed by the matrix structure descriptor (MSD) **1046** to the raw data matrix (RDM) **1042** property pane. Additionally, the matrix structure descriptor (MSD) **1046** provides processes required by application/database definition router (ADDR) **1028** and data link insertion component (DLI) **1044** for data field mapping and gating of vectorized access to individual data fields between objects, applications and databases.

[0153] Examples of enabling code for the raw data matrix property pane (RDM) **1042** and matrix structure descriptor (MSD) **1046** interface follow.

EXAMPLE 6

[0154] The raw data matrix property pane (RDM) **1042** contains such as data routing, data source type, data type definition, data content attributes and comment information. It also provides a mechanism linked to the matrix structure descriptor interface (MSD) **1046** for data authenticity and integrity checking.

```
    ///{{AFX_DATA_MAP(CPageRdmDlg)
    DDX_Control(pDX, IDC_DIALOGSIZE, m_buttDialogSize);
    DDX_Text(pDX, IDC_FILEPATH, file_path);
    DDV_MaxChars(pDX,file_path, _MAX_PATH);
    DDX_Text(pDX, IDC_DATASOURCETYPE, m_strDataSource);
    DDX_Text(pDX, IDC_CONTENTATTRIB, m_strContentAttrib);
    DDX_Text(pDX, IDC_COMMENTS, comments);
    DDV_MaxChars(pDX, comments, 320);
    //}}AFX_DATA_MAP
. . . . .
  m_nDataSource = sf.GetSourceType();
  ASSERT(m_nDataSource >= 0);
  switch(m_nDataSource)
  {
    case 0: m_strDataSource = _T("File");     break;
    case 1: m_strDataSource = _T("Application"); break;
    case 2: m_strDataSource = _T("Database");    break,
    case 3: m_strDataSource = _T("Instrument"); break;
    default: ASSERT(FALSE);
}
```

[0155] If raw data of any type are assigned to the Intelligent Object (IMO) **200** the first time, an integrity check is performed and a verifier created.

```
//set data integrity check
switch(m_nDataSource)
{
      case 0:
      {
            UUIDWSTR uuid_str = {0};
            StringFromGUID2(doc->m_uuidObject, uuid_str, sizeof(uuid_str));
            CString obj_uuid = uuid_str;
            CString result = CRawCheckSum::CheckRawData(filename, uuid_str);
            ASSERT(result.GetLength() == 32);
            doc->m_strDataIntegrity = result;
            TRACE(_T("Integrity Check for %s is %s. Length = %d\n"),filename, result, result.GetLength());
            break,
      }
      case 1: case 2: case 3:
      default: ASSERT(FALSE);
}
doc->m_strIcr = filename;
UpdateData(FALSE);
m_bDataLoaded = TRUE;
if((m_bDataLoaded == TRUE) && (m_bContentAttribLoaded == TRUE))
      doc->m_bHasRawData = TRUE;
AddToStateHistory();
doc->GenerateThumbnail();
doc->SetModifiedFlag(),
doc->UpdateAllViews(NULL);
}
(End Example 6)
```

[0156] An automatically or user evoked dynamic load function provides automated dynamic loading of Intelligent Objects (IMO) **200** containing similar content including a graphical preview of the Intelligent Object's (IMO) **200** content.

[0157] An optional text annotation property pane (TAN) **1038** provides a location for customized text annotation, integrated text viewing and definition and integration of links to external textual resources.

[0158] An optional text annotation interface (TANi) **1040** provides processing for customized text annotation, integration of linked data for viewing, formatting and support for integration of links to external text resources. Text annotation and viewing can be enabled by linking directly to the user's text editor of choice, by translation of text via external components to meet existing application requirements, or by calling the required text annotation application for viewing and editing of specified text. Additionally, the text annotation interface (TANi) **1040** provides an interface for text mining algorithms provided by external software components for distributed learning and knowledge extraction.

[0159] Additional optional property panes and their related processing components and access interfaces not shown in this exemplary embodiment, may include, for example, a knowledge extraction engine property pane comprising links to a set of "intelligent" algorithm for automated text and data searching, analysis and report generation.

[0160] It is evident from the above description, that this Intelligent Object (IMO) **200** architecture allows for real-time answers to complex, multidimensional, interdependent queries by providing the infrastructure for a global, comprehensive analysis of otherwise not accessible vast, inconsistent data sets.

[0161] Although the foregoing invention has been described in some detail by way of illustration and example for purposes of clarity of understanding, it will be readily apparent to those of ordinary skill in the art in light of the teachings of this invention that certain changes and modifications may be made thereto without departing from the spirit or scope of the appended claims.

[0162] The following examples are offered by way of illustration and not by way of limitation.

TABLE 1

OBJECT STATE TABLE
Tree branching flag bits:
0 = nothing
1 = beginning of the section (open tree branch)
2 = section entry
4 = last section entry (close tree branch)
Validation flag bits:
8 = Alert - Red
16 = Warning - Orange
32 = Approved - Green
000 ("Creation")
001 ("Object Created")
002 ("Sample Recorded")
005 ("Data Insert From File")

TABLE 1-continued

OBJECT STATE TABLE
006 ("Data Insert From Application")
007 ("Data Insert From Database")
010 ("Data Acquisition")
011 ("In Progress")
012 ("Interrupted")
013 ("Continued")
014 ("Completed")
020 ("Data Type Definition")
021 ("Raw")
022 ("Matrix")
023 ("Structure")
024 ("Vector")
025 ("Pointer")
026 ("Path")
027 ("Binary")
028 ("ASCII")
030 ("Database Record")
035 ("Relative Calibrations")
036 ("Relative Migration Distance (Rf)")
037 ("Relative Isoelectric Point (RpI")
038 ("Relative Mass (Rm)")
039 ("Relative Mobility (u)")
040 ("Biological Calibrations")
041 ("Concentration (Quantitation)")
042 ("Retention Time (Rt)")
043 ("Molecular Mass")
044 ("Molecular Size")
045 ("Isoelectric Point (Charge)")
046 ("Enzyme Activity")
047 ("Immunological Activity")
048 ("Other Bio-Activity")
050 ("Image Calibrations")
051 ("Optical Density (OD)")
052 ("Fluorescence Intensity")
053 ("Luminescene Intensity")
054 ("White Balance")
055 ("Light Intensity")
056 ("Dynamic Range Verification")
060 ("Measurement Calibrations")
061 ("Origin")
062 ("Size X")
063 ("Size Y")
064 ("Position X")
065 ("Position Y")
066 ("Position Z")
067 ("Length")
068 ("Width")
069 ("Depth")
070 ("Annotations")
071 ("Descriptive Text")
072 ("Base Designation (DNA Sequence)")
073 ("Aminoacid Designation (Protein Sequence)")
074 ("Molecular Type")
075 ("Molecular Function")
076 ("Molecular Structure")
077 ("Clinical Trial Information")
080 ("Functionality Information")
081 ("Organism")
082 ("Molecular Classification")
083 ("Metabolic Phase")
084 ("Disease Relationship")
090 ("Metadata")
091 ("Metadata Index Generated")
092 ("Metadata Index Updated")
100 ("Standardization/Normalization")
101 ("Threshold")
102 ("Base Function")
103 ("Base Plane")
104 ("Relative Location")
300 ("Experiment Validation")
301 ("Invalid Conditions")
302 ("Invalid Experiment")
303 ("Invalid Interpretation")
304 ("Invalid Analysis Tool Applied")

TABLE 1-continued

OBJECT STATE TABLE
306 ("Raw Data Integrity Verified")
307 ("Raw Data Integrity Violated")
308 ("Unsigned Raw Data")
310 ("Unsure Result")
311 ("Unsure Conditions")
312 ("Unsure Method")
315 ("Approved Method")
316 ("Certified Method")
320 ("Validated Method")
321 ("Validated Analysis")
325 ("Validated Annotation")
330 ("Validated Experiment")
340 ("Forensic Experiment")
350 ("Certified Experiment")
400 ("Output Request")
410 ("Numerical Output Generated")
411 ("Graphical Output Generated")
420 ("Remote Output Sent")
430 ("Output Printed")
900 ("Access Information")
901 ("Access Denied")
902 ("Time Limit Exceeded")
903 ("User Limit Exceeded")
910 ("Inter-Object Communication Granted")
911 ("User Access Granted")
920 ("Full Access Granted")
930 ("Limited Access Granted")
940 ("Local Access Granted")
941 ("Local Access Denied")
942 ("Local Access Restricted")
950 ("Remote Access Granted")
951 ("Remote Access Denied")
952 ("Remote Access Restricted")

[0163]

TABLE 2

CONTENT ATTRIBUTE LIST
Tree branching flag bits:
0 = nothing
1 = beginning of the section (open tree branch)
2 = section entry
4 = one up
8 = two up
16 = three up
000 ("Life Sciences"),
010 ("Experimental Planning"),
011 ("Scope"),
012 ("Requirements"),
013 ("Resources And Costs"),
015 ("Materials And Methods"),
020 ("Instrumentation"),
030 ("Experiment Optimization"),
050 ("Literature Services"),
051 ("ISIS"),
052 ("Current Contents"),
060 ("Medline"),
100 ("Sample Information"),
101 ("Sample Origin"),
102 ("Sampling Technique"),
103 ("Organism"),
104 ("Type"),
105 ("Description"),
106 ("Storage Requirements"),
107 ("Amount Available"),
130 ("Sample Preparation"),
131 ("Selective Dissolution"),
145 ("Solvent Extraction"),
150 ("Acetone Dry Powder"),
200 ("Analytical Methods And Tools"),
201 ("Atomic Absorption"),
202 ("Aminoacid Analysis"),
203 ("Chromatography"),
204 ("DC, TLC"),
205 ("LC, HPLC"),
206 ("GC"),
209 ("Electrophoresis"),
210 ("Electrophoresis, 1DB Gels, Native"),
211 ("Visible stains"),
212 ("Silver"),
213 ("Coomassie Blue"),
214 ("Amido S"),
215 ("Fast Green SF"),
216 ("Auto Radiography"),
220 ("Fluorescent stains"),
221 ("Ethidium Bromide"),
222 ("TOTO/YOYO"),
223 ("SYPRO Orange"),
224 ("SYPRO Ruby"),
225 ("SYBR Green"),
226 ("DAPI"),
230 ("Electrophoresis, 1DE Gels, Denaturing"),
231 ("Visible Stains"),
232 ("Silver"),
233 ("Coomassie Blue"),
234 ("Amido 5"),
235 ("Fast Green SF"),
236 ("Auto Radiography"),
240 ("Fluorescent Stains"),
241 ("Ethidium Bromide"),
242 ("TOTO/YOYO"),
243 ("SYPRO Orange"),
244 ("SYPRO Ruby"),
245 ("SYBR Green"),
246 ("DAPI"),
250 ("Electrophoresis, 1DE IEF, Ampholytes"),
251 ("Visible stains"),
252 ("Silver"),
253 ("Coomassie Blue"),
255 ("Fluorescent stains"),
256 ("Ethidium Bromide"),
257 ("TOTO/YOYO"),
260 ("Electrophoresis, 1DB IPG, Immobilzed"),
261 ("Visible stains"),
262 ("Silver"),
263 ("Coomassie Blue"),
265 ("Fluorescent stains"),
266 ("Ethidium Bromide"),
267 ("TOTO/YOYO"),
270 ("Electrophoresis, MADGE"),
271 ("Visible stains"),
272 ("Silver"),
273 ("Coomassie Blue"),
275 ("Fluorescent stains"),
276 ("Ethidium Bromide"),
277 ("TOTO/YOYO"),
278 ("SYPRO Ruby"),
280 ("Electrophoresis, 2DE Gels, Ampholyte Focused"),
281 ("Visible Stains"),
282 ("Silver"),
283 ("Coomassie Blue"),
290 ("Fluorescent Stains"),
291 ("Ethidium Bromide"),
292 ("TOTO/YOYO"),
293 ("Cy Dyes"),
294 ("SYPRO Orange"),
300 ("Electrophoresis, 2DE Gels, IPG Focused"),
301 ("Visible Stains"),
302 ("Silver"),
303 ("Coomassie Blue"),
310 ("Fluorescent Stains"),

TABLE 2-continued

CONTENT ATTRIBUTE LIST
311 ("Ethidium Bromide"),
312 ("TOTO/YOYO"),
313 ("Cy Dyes"),
315 ("Electrophoresis, CE"),
316 ("UV/VIS Detection"),
317 ("Fluorescence Detection"),
318 ("Other Detection"),
320 ("Spectroscopy"),
321 ("UV/VIS"),
322 ("Infrared (IR)"),
323 ("NMR"),
324 ("ESR"),
325 ("Mass Spectrometry (MS)"),
326 ("MALDI-ToF"),
327 ("SELDI"),
328 ("Quadrupol"),
330 ("Combined MS Techniques"),
331 ("LC-MS"),
335 ("CE-MS"),
340 ("MS-MS"),
350 ("Sequencing"),
351 ("Nucleic Acids"),
352 ("Proteins"),
353 ("Glycopeptides, Carbohydrates"),
360 ("Kinetics"),
361 ("Enzymes"),
362 ("Substrates"),
365 ("Metabolites"),
370 ("Immunoturbidimetry"),
375 ("Microbiology"),
376 ("Colony Counting"),
377 ("Colony Differentiation"),
380 ("Bioassays"),
381 ("Antibiotics Inhibition"),
385 ("Immunoaffinity"),
386 ("Binding Coefficients"),
390 ("Chemical Structure"),
391 ("X-ray Crystallography"),
395 ("Microscopy"),
396 ("Light/Polarisation Microscopy"),
397 ("Fluorescence Microscopy"),
398 ("Electron Microscopy"),
400 ("Macro/Micro Arrays"),
401 ("Microtiter Plates (96/384 Wells)"),
402 ("Agar Punchplates (6 × 6/8 × 8/12 × 12 Holes"),
405 ("CHIP-based Arrays"),
406 ("High Density Arrays"),
410 ("Screening"),
460 ("Synthesis"),
500 ("Drug Discovery"),
600 ("Drug Development"),
680 ("Toxicology"),
700 ("Regulatory Compliance"),
701 ("Pre-clinical Trial"),
710 ("Clinical Trial, Phase I"),
720 ("Clinical Trial, Phase II"),
730 ("Clinical Trial, Phase III"),
740 ("Post-Trial Compliance Compilation"),
750 ("QC/QA: Reproducibility"),
780 ("Method Reliability Tests"),
781 ("Accuracy"),
782 ("Repeatability"),
800 ("Generic Statistics Tools"),
801 ("Correlation"),
802 ("Regression"),
810 ("Pairwise Comparison"),
820 ("Similarity Clustering (Dendrograms)"),
830 ("Experiment Congruence (SimPlots)"),
840 ("Principal component Analysis"),
850 ("Self-organizing Map (SOM)"),
900 ("Modeling And Prediction"),
901 ("Separation Profile Forecast"),
910 ("Structure-Based Function"),
920 ("Clinical Trial Simulation"),

We claim:

1. A data structure ("Intelligent Molecular Object, hereinafter "Intelligent Object") providing methods for unified, functional integration and unified data content access and management, wherein the methods comprise:

a) at least one component comprising procedures for persistent data content and activity status management; and

b) at least one interface comprising procedures for functional integration and information interchange between the data, associated data and applications, components and interfaces.

2. An Intelligent Object as in claim 1, wherein the procedures for persistent data content and activity status management further comprise

a) at least one component comprising procedures for persistent data content and activity status management across diverse data types, content and computing environments; and

b) at least one interface comprising procedures for direct data content addressing, definition and presentation of data content properties for information interchange between said data, associated data and applications, components and interfaces across diverse data types, content and computing environments.

3. An Intelligent Object as in claim 1, wherein said procedures for persistent data content and activity status management across diverse data types, content and computing environments further comprise

a) at least one set of procedural steps for data content access and processing synchronization;

b) at least one set of procedural steps for persistent data content access and processing logging and auditing;

c) at least one set of procedural steps for monitoring and verification of data integrity, command history, and regulatory validation states.

4. An Intelligent Object as in claim 1, wherein said interface procedures for direct data content addressing, definition, presentation and functional integration further comprise methods for:

a) vectorized access, routing, translation, linking and comparison of data content information such as data content and data content subsets ("workspaces") defined to granularity from entire data content files to data content subsets as small as single byte workspaces;

b) and direct interchange of information not limited to said not limited to processing information, commands, queries, meta-data, data content and said defined workspace subsets;

i. between said data objects; and

ii. between said data objects, heterogeneous applications and a variety of components, access interfaces, and presentation layers.

5. An Intelligent Object as in claim 1, further comprising:

a) a set of unified presentation layers ("property panes") comprising methods for user viewing and interactivity;

b) a set of unique components (and/or "processing components") comprising methods for raw data matrix mapping, status management, data access and routing, data content translation and/or activation of external processing components for data processing; and

c) a set of functional interface layers ("access interfaces") comprising methods for direct data content comparison and information interchange between addressed data content, meta-data, applications and components.

6. A data structure as in claim 1, further comprised as a core element within an information technology platform architecture ("Sentient Platform") advantageously enabled in software, comprising methods for:

a) functional data integration, multidimensional data accessing and routing, viewing, querying, analyzing and other data-enabling operations

i. utilizing single data content files and/or high numbers of complex, heterogeneous and/or interdependent data content files which are functionally unified, although utilizing diverse and/or distributed data content resources; and

ii. capable of simultaneously utilizing data content stored in diverse and distributed database and/or data storage resources, diverse computational hardware and network environments with diverse and dynamic application needs.

7. The creation method of said data structure, which comprises methods and processes not limited to

a) automated detection of data content newly available locally and/or over networks that correspond to actions including user initiated and/or automated queries, user initiated and/or automated data acquisition activities and/or data import requests

b) invocation of a new unique object identifier property pane via a component comprised within an external object handler,

c) assignation of a globally unique identifier,

d) generation of functionally integrated property panes within the Intelligent Object;

e) identification of all user-defined and content attribute based connections to the data structure; and

f) listing within the unique object identifier property pane of information comprising the Intelligent Object's creator, network and/or local routing information, content attribute information, information on connected users, access permissions, authentication information such as globally unique identification and public key encryption status, and current session information; and

g) state history recordation of all Intelligent Object activity over networks and/or within local computing environments.

8. A unique object identifier property pane (UID), for interactive user, data, session and machine authentication, comprising methods for

a) bi-directional information interchange with an external application for data handling and applications integration ("IOH") and a comprised unique object identifier access interface; and

b) provision of globally unique identification of said Intelligent Object utilizing globally unique identifiers comprising an 128-bit alphanumeric string.

9. A unique object identifier property pane as in claim 8, wherein the procedures for identification further comprise methods for:

a) provision of additional identification information including;

i. data creation information, comprising time, date and machine identification information;

ii. ownership information, comprising unique identification information for the object creator;

iii. security setting information, comprising access permissions, read-only permission, write permission;

iv. routing information, comprising raw data content addressing information and Intelligent Object data addressing information; and

v. real-time information about locally or remotely connected users, comprising user's unique identification information; computer name; login time; network address or location information; and session identification information; and

b) at least one rule or policy for object data security which either permits or denies access for information interchange between data objects, data content, users and applications, based on comprised criteria for session, user and data authentication.

10. An interactive content router component (ICR), for addressing of data content access and routing, comprising methods for

a) bi-directional information interchange with components and access interfaces including said object root router component, a status management component, an object pane descriptor component, and a unique object identifier Access Interface; and

b) utilization of information provided by said object pane descriptor component and comprised meta-data index information to optimize linking, organization and direction of vectorized access and routing requests, processing commands, and information interchange between such as Intelligent Objects, heterogeneous and/or homogeneous data objects and data content, applications, data resources and/or databases.

11. An interactive content router component as in claim 10, further comprising methods for

a) implementation of read-only access to data content, and routing of said vectorized data to an external application for data handling, for cache-based processing according to non-destructive overlay methods;

b) addressing to multiple data content locations according to provided criteria;

c) direction of said vectorized data content access and routing within any local computing environment, local network, wide area network and/or global network;

d) direction of meta-data, data content and/or processing results routing within said computing environments;

e) Additionally, said comprised methods and processes report data content origination and routing information to a unique object identifier access interface.

12. An object root router component (ORR), for addressing, accessing and routing of the Intelligent Object, comprising methods for

a) bi-directional information interchange with components and access interfaces such as a unique object identifier access interface, an interactive content router component, and a status management component;

b) definition of the origin of the Intelligent Object; and

c) addressing for access and routing of the Intelligent Object.

13. An object root router component as in claim 12, further comprising methods for

a) definition of the origin of the Intelligent Object and for multiple addresses to said Intelligent Object, locally and/or over any local network and/or global network;

b) definition of the origin of associated data objects according to object description and relationship definitions; and

c) addressing for data management, accessing and routing of the Intelligent Object within local, distributed and/or remote database and data storage technology.

14. A unique object identifier access interface (UIDi), for authentication, validation, and gating permission or denial of access to said Intelligent Object and comprised data content, comprising methods for

a) bi-directional information interchange with components and access interfaces including an object root router component, interactive content routing component, status management component and said unique object identifier property pane; and

b) permission or denial of access to said elements, components and access interfaces.

15. A unique object identifier access interface as in claim 14, further comprising methods for

a) fielding of electronic signatures for validated identification and authentication.

16. An object state engine property pane (OSE), for viewing and interactivity regarding Intelligent Object state information, comprising methods for

a) bi-directional information interchange with components and access interfaces including an external object handler user interface and a status management component; and

b) interactive, real-time Intelligent Object activity viewing, activity alerting, auditing and manual activity synchronization.

17. An object state engine property pane as in claim 16, further comprising methods for

a) detailed activity logging via a linked status management component of state attributes including: globally unique user identification; computer, instrument or device identification; login date and time; logout date and time; user; Intelligent Object and data content addressing information; unique session identification; data authentication information; data integrity information; user access authentication and/or denial of access; session activity records; data ranking status; and/or validation states.

18. An always-on status management component (SMC), for persistent data object state history and status management, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to said object root routing component, said interactive content routing component, said unique object identification access interface, said object state engine property pane, an object graph preview component, an object pane descriptor component, and an external object state engine; and

b) launching of and interaction with said external object state engine component.

19. An always-on status management component as in claim 18, further comprising methods for

a) communication with an external object state engine to enable monitoring of data integrity, command history and regulatory validation states;

b) communication with an external object state engine to enable viewing of instrument parameters related to acquisition of data content and/or other instrument activities

c) communication with said object state engine to provide automated validation status verification according industry-specific validation requirements not limited to Good Practice (G*P)-compliance including GLP Laboratory Information Management Systems (LIMS); U.S. Food and Drug Administration Center for Drug Evaluation and Research (FDA CDER); Codebook for Forensic Investigation Requirements (CFIR); and International Organization for Standardization (ISO 9000); and

d) provision of information including data acquisition state; calibration information; applied transformation or analysis processes; and validation state (Object State List—Table 1) to the object state engine property pane;

e) utilization of comprised state history information and non-destructive overlay processing techniques for data integrity protection and verification; auditing; logging; and detailed rollback to previous Intelligent Object data states; and

f) functional integration of data and applications via interactive information exchange with external components that provide processes including data type translation and applications integration.

20. An object query interface access interface (OQI), for massively parallel and direct Intelligent Object-to-Intelligent Object information interchange, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to an object pane descriptor component, a meta-data index access interface, and said status management component;

b) reception of query requests from the status management component and an external object state engine component;

c) implementation of queries across data subsets comprised within homogeneous and/or heterogeneous data content storage resources, based on query information provided by the object pane descriptor component, meta-data index interface and status management component;

d) initiation of direct object-to-object information interchange in parallel with corresponding Intelligent Objects via their comprised object query interfaces, by directing the interactive content routing of specified data vectors and/or meta-data index information directly between object query interfaces and of other Intelligent Objects for linking, comparison and relationship definition;

e) relaying said data content information to an external result aggregation engine for further processing and presentation to the user via the unified presentation layer user interface; routing of results of Boolean comparisons and other algorithms applied to linked data content and/or meta-data compared back to the object pane descriptor property pane for presentation to the user; and

f) routing of results of said direct object-to-object information interchange to

i. update an object pane descriptor component and

ii. update a meta-data index access interface.

21. An object query interface access interface as in claim 20, further comprising methods for

a) implementation of queries across data subsets comprised within an external object data storage resources, based on query information provided by the object pane descriptor component, meta-data index interface and status management component;

22. A meta-data index property pane (MDX), for unified user viewing and interactivity with data object meta-data, comprising methods for:

a) bi-directional information interchange with components and access interfaces not limited to an external application for data handling and applications integration and a meta-data index access interface;

b) viewing of meta-data index information relating to said Intelligent Object and its data content and activity, said information comprising automatically generated and/or user-defined meta-descriptions and query relationship trees regarding information comprising specific data functionality and/or relationships to other data and/or data inter-dependencies.

23. A meta-data index property pane as in claim 22, further comprising methods for:

a) interactive user definition and updating of said meta-data index information relating to said Intelligent Object and its data content and activity.

24. A meta-data index access interface (MDXi), for functional integration of data object meta-data according to user activities, data properties and analytical requirements, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to said object query interface, said meta-data index property pane, and an object pane descriptor component;

b) dynamic integration of meta-data information resulting from such as user-based and/or automated query histories, various applied analytical command and result histories, user-defined meta-data entries, and/or a combination thereof; and aggregation and integration of results of multi-parametric clustering and/or other data analyses; and

c) provision of sets of rules to optimize access and routing and by said Intelligent Object.

25. A meta-data index access interface as in claim 24, further comprising methods for

a) extraction, integration and implementation of a variety of result aggregation, distributed learning and knowledge extraction algorithms, such as but not limited to external result aggregation engines; distributed learning engines; and knowledge extraction engines.

26. An object pane descriptor property pane (OPD), for unified user viewing and interactivity with data object properties presented within the entity of property panes, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to an external application for data handling and applications integration and an object pane descriptor component; and

b) provision of an overview of properties presented within said Intelligent Object's property panes such as, but not limited to data content attributes, data state, data location and/or locations, data environment definitions, data structure and/or structures, meta-data, locally available and/or remotely linked applications, processing components and access interfaces, linked databases and/or data resources; and

c) provision of graphical data content and information views of linked data and applications.

27. An object pane descriptor property pane as in claim 26, further comprising methods for

a) viewing of information highlighting functional relationships between Intelligent Object data properties, and related applications and data; and

b) interactive "drag and drop" functionality to enable user definition of custom and/or automated analytical requirements.

28. An object pane descriptor component (OPD), for functional integration of data object properties presented within the entity of property panes, comprising methods for

a) bi-directional information interchange with components and access interfaces including said interactive content routing component, said status management component, said object query interface, said meta-data index access interface, said object pane descriptor property pane, an object graph preview property pane, an object graph preview access interface, an object access manager component, an applications/database definition router access interface, and a matrix structure descriptor component; to provide

b) dynamic detection, updating, routing and presentation of information required for functionality defined by the entity of object property panes comprised by said Intelligent Object; and

c) direct linking of data with functionally related Intelligent Objects, data content and related applications; to enable

d) unified analysis of data from homogeneous and/or heterogeneous data types from homogeneous and/or heterogeneous sources.

29. An object pane descriptor component as in claim 28, further comprising methods for

a) provision of an overview of properties presented within said Intelligent Object's property panes, such as data content attributes, data state, data location and/or locations, data environment definitions, data structure and/or structures, meta-data, locally available and/or remotely linked applications, processing components and access interfaces, linked databases and/or data resources

b) provision of information regarding functional relationships of data objects, data content and meta-data.

30. An object pane descriptor component as in claim 28, further comprising methods for

a) provision of functionally related data object, data content and meta-data information comprising: data content attributes; data content interrelationships; data state; data location and/or locations; data environment definitions; data structure and/or structures;

b) provision of said information for functional linking, vectorized accessing and routing of data content, meta-data and processing results;

c) functional integration of said data within a computing environment comprising locally available and/or remotely linked applications; processing components; access interfaces; databases and/or data resources located within local computing environments, local networks, wide area networks and/or global networks; and

d) viewing and interactive user definition and updating of said information to provide integrated functionality of properties defined by the entity of object property panes.

31. An applications/database definition router interface (ADDR), for definition and provision of application and database dependencies, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to said object pane descriptor component and a matrix structure descriptor interface;

b) addressing and functional linking of data content for interactivity with heterogeneous applications, databases and/or data resources.

32. An applications/database definition router interface as in claim 31, further comprising methods for

a) detection, extraction and definition of required and/or available data content attributes, data state, data locations, data environment definitions, data structures required for interactivity with locally available and/or remotely linked applications, databases and/or data resources.

33. An object graph preview property pane (OGP), for unified user viewing of data content, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to an external application for data handling and applications integration user interface, said status management component, said object pane descriptor component and an object graph preview interface;

b) user reviewing of data content such as low-resolution graphical "thumbnail" data previews.

34. An object graph preview property pane as in claim 33, further comprising methods for

a) interactive user identification of data content subsets for accessing and user reviewing.

35. An object graph preview property pane as in claim 33, further comprising methods for

a) Interactive user identification of data content for access according to meta-data and vectorized data access protocols; and

b) user reviewing of heterogeneous data content within a unified presentation environment comprising methods for detailed audio, graphic, digital video, review.

36. An object graph preview property pane as in claim 33, further comprising methods for

a) definition and selection of workspace subsets, which may include an entire data content file or a subset of said data content as small as a single byte, to applications such as pattern detection, normalization, comparison, 2D and 3D viewing and/or analysis;

b) interactive linking to external methods provided for detailed data relationship viewing, such as a data pool viewer; and

c) viewing of subsets of data content and/or linked data with corresponding content attributes.

37. An object graph preview interface (OGPi), for accessing and directing graphical data content, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to said object pane descriptor component, said object graph preview property pane, and an object access manager component;

b) routing and linking of data content between applications windows; and

c) linking of limited resolution image/graphics ("thumbnail") views of data content to the object graph preview property pane.

38. An object graph preview interface as in claim 37, further comprising methods for

a) launching of functionally integrated applications according to correspondence of data content, queries and applications functionality via a comprised module launch component (MLC);

b) routing and linking of data content including content attribute information, data annotations, limited resolution graphic views of any selected data file content, and

c) linking of limited resolution image/graphics ("thumbnail") views of data content from external applications and/or software systems such as operating systems, for passing said view to the object graph preview property pane.

39. An application translator link property pane (ATL), for interactive user modeling, linking, assembly, and integration of applications, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to an external application for data handling and applications integration and an object access manager component;

b) provision of a list of available and applicable applications, application components and related data resources; and

c) manual launching of external modules for cached non-destructive analysis and/or processing of data.

40. An application translator link property pane as in claim 39, further comprising methods for

a) provision of a list of available and applicable applications, analytical components and related data resources;

b) manual linking of data content to related software modules and/or processing devices such as preferred analytical applications, processing devices and other interactive devices for actions not limited to viewing, analyzing and/or listening; and

c) manual launching of external components for analysis and/or processing of data.

41. An application translator link property pane as in claim 39, further comprising methods for

a) provision of a user interface for unified application and application component relationship definition, functional access, organization, translation, and integration; and

b) interactive, unified user viewing of and interactivity with applications components and their required and/or available relationships;

i. depending on data content, functional data relationships, analytical requirements, available applications, available applications components, translation requirements and specific user queries;

c) provision of an interactive user interface for automated, instantaneous applications functionality integration via linking to an external component for automated applications assembly, to provide the assembly functionality for dynamic development of new and customized applications.

42. An object access manager component (OAM) for functional analytical application, component, interface and data integration, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to said object pane descriptor component, said object graph preview property pane and an object access manager component; and

b) defining parameters required for functional access, routing and presentation of data content within a given application or database environment.

43. An object access manager component, as in claim 42, further comprising methods for

a) provision of a messaging queue; which

i. defines the applications window content required for data content transfer between the Intelligent Object and heterogeneous applications;

ii. governs functional presentation within homogeneous and/or heterogeneous application and database environments; and

iii. functionally integrates data content with a variety of applications;

b) direction of access and routing requests for specific data content to linked applications;

c) direction of information interchange between the Intelligent Object and external processing components and/or various applications such as text editors to allow for analysis, customization, formatting, processing, reviewing and unified presentation of said data content in a non-destructive manner, through said external components and applications;

d) interaction with external object translation engines to provide

i. data content translation for functional integration with a variety of applications; and

ii. data content and attribute-based algorithms for applications translation, integration and inter-application communication;

e) direct linking of data content and meta-data to external components comprising methods including data content translation and presentation according to requirements of heterogeneous software modules or plug-ins;

f) direct linking of data content and meta-data to external applications including applications running on handheld electronic devices such as those used for data text entry; and

g) direct linking of data content and meta-data to external applications including laboratory instrument plug-ins for real-time viewing of instrument and/or data parameters related to acquisition of data content.

44. A raw data matrix property pane (RDM), for provision of detailed data storage and access information, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to an external application for data handling and applications integration user interface and a data link insertion component;

b) provision of information regarding originating data content matrices.

45. A raw data matrix property pane as in claim 44, further comprising methods for

a) user reviewing of data content information such as data source location, data source type, data storage information, data access; data content path and vectorized data matrix structure information;

b) provision of information regarding originating data content matrices, across heterogeneous originating data and data resource type and/or structure.

46. A data link insertion component (DLI), for synchronization of data content access and routing with required transport protocols, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to said raw data matrix property pane; and matrix structure descriptor access interface;

b) data content distinction by type membership such as file, database record, application, and live data acquisition; and

c) detection and activation of transport protocols required for specified data type accessing and routing requests.

47. A data link insertion component as in claim 46, further comprising methods for

a) detection and definition of data type, transport and routing protocols for the required communication mechanism to data content such as PPP over TC/IP, peer-to-peer, client/server, server/server, http and/or ftp.

48. A matrix structure descriptor component (MSD), for provision of vectorized data content mapping, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to said status management component, said object pane descriptor component; said Application/Database Definition Router interface; and said data link insertion component;

b) matrix structure description of data content according to comprised vector mapping definitions; and

c) mapping of data according to data field, data format and/or data structure.

49. A matrix structure descriptor component as in claim 48, further comprising methods for

a) linking of said data matrix structure information to meta-data and data description information;

b) linking of addressing definitions required to enable functional integration data content accessing; and

c) accessing of data content ("workspaces") to levels of granularity including data subset overviews, data files, fields, rows, cells, pixels or bytes;

50. A text annotation property pane (TAN), for unified, interactive user text entry and linking of textual and other references, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to an external application for data handling and applications integration user interface and a text annotation access interface;

b) manual entry, viewing and linking of information such as text annotation;

c) activation of external components or applications such as text editors to allow for customization, formatting, reviewing and processing of said information through external editors; and

d) provision of functionality for linking of customized text annotations, external text resources, references, related data and information.

51. A text annotation access interface (TANi), for direct linking of external applications for text editing and for linking of textual and other reference files, comprising methods for

a) bi-directional information interchange with components and access interfaces not limited to said object access manager; and said text annotation property pane;

b) linking of external components or applications such as text editors to allow for customization, formatting, reviewing and processing of said information through external editors; and which allow for

c) passing of said information back to the text annotation property pane.

* * * * *