(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 January 2005 (27.01.2005)

PCT

(10) International Publication Number
WO 2005/008536 A1

(51) International Patent Classification[7]: G06F 17/30

(21) International Application Number: PCT/US2004/022031

(22) International Filing Date: 8 July 2004 (08.07.2004)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/486,779	11 July 2003 (11.07.2003)	US
10/887,137	7 July 2004 (07.07.2004)	US

(71) Applicant *(for all designated States except US)*: **COMPUTER ASSOCIATES THINK, INC.** [US/US]; One Computer Associates Plaza, Islandia, NY 11749-7000 (US).

(72) **Inventor; and**
(75) **Inventor/Applicant** *(for US only)*: **VAUGHT, Jeffrey, A.** [US/US]; 4107 Woodmont Drive, Batavia, OH 45103-2567 (US).

(74) **Agent: STALFORD, Terry, J.**; Fish & Richardson P.C., 5000 Bank One Center, 1717 Main Street, Dallas, TX 75201-4605 (US).

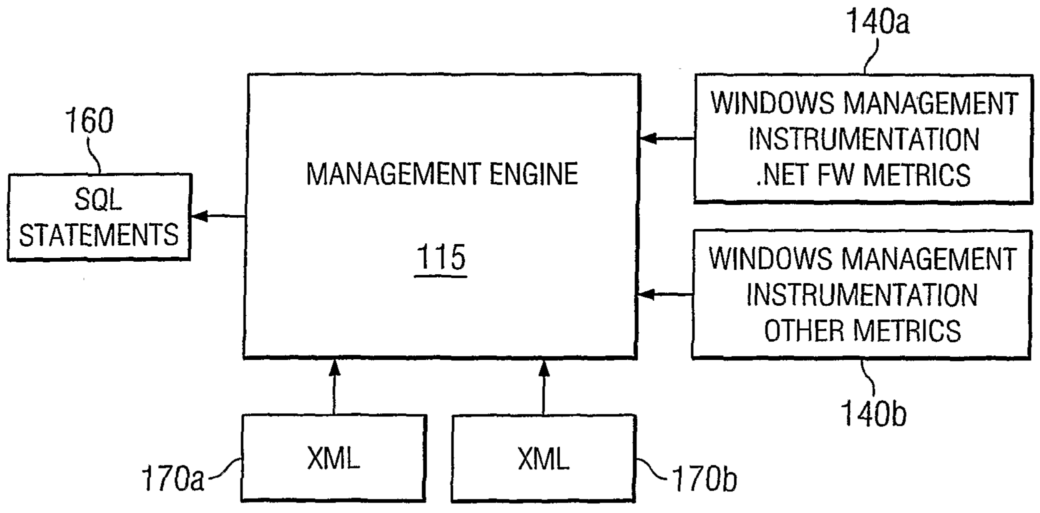
(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) **Title:** SYSTEM AND METHOD FOR STORING METRICS IN A DATABASE


140a
160
SQL STATEMENTS
MANAGEMENT ENGINE
115
WINDOWS MANAGEMENT INSTRUMENTATION .NET FW METRICS
WINDOWS MANAGEMENT INSTRUMENTATION OTHER METRICS
140b
XML
XML
170a
170b

(57) **Abstract:** A system and method for storing data associated with an extensible instrumentation layer are provided. The method includes receiving metrics (140) from an extensible instrumentation layer (130) in an operating system (110). The metrics (140) are defined by at least one class and a plurality of properties, with each property being associated with one class. The metrics (140) are converted into at least one database-compliant data structure and stored in a database (124).

WO 2005/008536 A1

SYSTEM AND METHOD FOR STORING METRICS IN A DATABASE

REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. provisional application serial number 60/486,779 filed July 11, 2003 entitled, "SYSTEM AND METHOD ASSOCIATING CLASSES AND PROPERTIES WITH DATABASE TABLES" and the U.S. utility application filed July 7, 2004 entitled, "SYSTEM AND METHOD FOR STORING METRICS IN A DATABASE".

TECHNICAL FIELD

This disclosure relates generally to the field of data processing and, more specifically, to storing metrics in a database.

BACKGROUND

Microsoft's .NET application is software that includes the .NET Framework, which is typically used for developing and running network-based applications and web services. These applications often operate over a network through standard, platform-independent protocols including, for example, eXtensible Markup Language (XML), Simple Object Access Protocol (SOAP), and Hypertext Transfer Protocol (HTTP). The .NET Framework includes the common language runtime (CLR), which assists memory, process, and thread management. The CLR Profiler is a profiling tool for .NET-compatible applications. Windows Management Instrumentation (WMI) is an extensible instrumentation layer built into many Windows or Windows-based operating systems. WMI exposes namespaces, classes, and properties for hardware devices, the operating system, and applications. When Microsoft .NET framework is installed, a number of classes are created for the management of .NET. These are often located within "root/CIMV2" and may include various CLR classes, ASP.NET Overall, and ASP.NET by Application, which normally includes web services. Moreover, many correlated operating systems metrics, such as CPU load and disk queue length, can be collected from the extensible instrumentation layer.

SUMMARY

A system and method for storing data associated with an extensible instrumentation layer are provided. In one embodiment, the method includes receiving metrics from an extensible instrumentation layer in an operating system. The metrics are defined by at least one class and a plurality of properties, with each property being associated with one class. The metrics are converted into at least one database-compliant data structure and stored in a database. The details of one or more embodiments of the disclosure are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the disclosure will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

FIGURE 1 illustrates a system for storing metrics in a database according to certain embodiments of the disclosure;

FIGURES 2A-B illustrate alternative configurations of the scaleable system in FIGURE 1;

FIGURE 3 is an example data flow diagram illustrating the mapping of metrics to database-compliant data structures;

FIGURES 4A-B illustrate various displays of example database tables storing Windows metrics; and

FIGURES 5A-B are flow diagrams illustrating example methods for storing Windows metrics in a database according to various embodiments of the disclosure.

DETAILED DESCRIPTION

FIGURE 1 is a block diagram illustrating a distributed computing system 100 including management engine 115 for storing Windows, Windows-based, or other class-based metrics 140 in a database-like repository according to one embodiment of the present disclosure. Metrics 140 may be any data or information collected or generated by an extensible instrumentation layer 130. For example, metrics 140 may include load time for one or more assemblies, modules, and classes, load count of one or more assemblies, modules, and classes, load failures for one or more classes, jit compile time for one or more functions, jit search time for one or more functions, jit count for one or more functions, jit compile failures for one or more functions, function execution time for one or more functions, function exceptions for one or more functions, function interop boundary crossings for one or more functions, number of compilations, number of sessions, number of applications, and many others. At a high level, system 100 is a client/server environment comprising at least one client or management workstation 104, a server or host 102, and network 108, but may also be a standard or local computing environment or any other suitable environment. In general, system 100 dynamically provides a system component or a system administrator or other user with a plurality of Windows, Windows-based, or Windows-compatible metrics 140, typically communicated in a class/method format, in a database table/column format. For example, system 100 may comprise an environment automatically providing one or more users with the ability to easily manage or view .NET framework metrics 140 and correlated metrics 140. The term "automatically," as used herein, generally means that the appropriate processing is substantially performed by at least part of system 100. It should be understood that "automatically" further contemplates any suitable user or administrator interaction with system 100 without departing from the scope of this disclosure. The term "dynamically," as used herein, generally means that certain processing is determined, at least in part, at run-time based on one or more variables.

Server 102 includes memory 120 and processor 125 and comprises an electronic computing device operable to receive, transmit, process and store data associated with system 100. Server 102 may comprise a general-purpose personal computer (PC), a Macintosh, a workstation, a Unix-based computer, a server computer, or any other suitable device. For example, server 102 may be a blade server or a web server. In short, server 102 may comprise software and/or hardware in any combination suitable to gather metrics 140 and convert metrics 140 into one or more database-compliant data structures. FIGURE 1 only provides one example of computers that may be used with the disclosure. For example, although FIGURE 1 provides one example of server 102 that may be used with the disclosure, system 100 can be implemented using computers other than servers, as well as a server pool. The present disclosure contemplates computers other than general purpose computers as well as computers without conventional operating systems. In other words, as used in this document, the term "computer" is intended to encompass any suitable processing device. Computer server 102 may be adapted to execute any operating system 110 including Windows NT, Windows 2000, Windows Server, Windows Storage Server, Windows XP home or professional, or any other suitable operating system including an extensible instrumentation layer 130.

Memory 120 may include any memory or database module and may take the form of volatile or non-volatile memory including, without limitation, magnetic media, optical media, random access memory (RAM), read-only memory (ROM), removable media, or any other suitable local or remote memory component. In this embodiment, illustrated memory 120 includes database 124, at least one mapping table 170, and (at least temporarily) one or more SQL scripts or statements 160, but may also include any other appropriate data. Database 124 stores one or more database tables, with each table including one or more columns. Database 124 may receive records, schemas, or any other suitable data through interface 112 or from another process running on server 102. In one embodiment, database 124 may be a relational database management system (or DBMS). Relational databases often use sets of schemas to describe the tables, columns, and relationships between the tables using basic principles known in the field of database design. But while described as a relational database, database 124 may be any data repository of any suitable format including XML documents, flat files, Btrieve files, comma-separated-value (CSV) files, an object oriented database, name-value pairs, and others so long as it remains operable to load, store, interface, or reference one or more SQL scripts 160.

Generally, SQL script 160 comprises any SQL API, code, or other statement 165 operable to process any suitable data storage. For example, SQL script 160 may comprise a plurality of SQL statements 165, such as JOIN, DROP_TABLE, MODIFY, SELECT, DELETE, or UPDATE, without departing from the scope of this disclosure. It will be understood that script 160 may include one or more SQL statements 165 and may be used interchangeably as appropriate without departing from the scope of the disclosure. SQL script 160 may be used by any DBMS or database 124, whether local or remote, to select, modify, delete, or otherwise process one or more data structures associated with database 124. As used herein, "SQL" references any of the plurality of versions of the SQL relational database query and manipulation language such as, for example, SQL, ANSI SQL, or any other variant or generic database or repository query language. Moreover, this disclosure contemplates any suitable API, script, or executable in any suitable language and format compatible with database 124.

Memory 120 also stores one or more mapping tables 170, with each mapping table 170 comprising a logical format operable to map, or is associated with, at least a subset of metrics 140. Each mapping table 170 may be an XML document, an object, an array, or any logical or physical component operable to map metrics 140 to a database schema using SQL script 160. Further, it will be understood that mapping table 170 may be local or remote, as well as temporary or persistent, without departing from the scope of the disclosure. In one embodiment, system 100 may include two mapping tables 170. For example, first mapping table 170 may be used by management engine 115 to map .NET framework metrics 140 and second mapping table 170 may be used by management engine 115 to map correlated or other metrics 140. Moreover, first and second mapping tables 170 may be two logical portions of the same XML document, object, or array. For example, mapping table 170 may include one or more portions, tags, and sub-tags in a readable format such as illustrated below:

.NET framework portion

```
        <DatabaseTbl InstTbl="WMI_CLRLoad_Instance"
ThreshTbl="WMI_CLRLoad_Thresh"
        WMIClass="Win32_PerfRawData_NETFramework_NETCLRLoading">
                <Prop DBCol="Node" Text="Y" ThreshTbl="Y"> </Prop>
                <Prop DBCol="ClrName" Text="Y" ThreshTbl="Y">Name</Prop>
                <Prop DBCol="TStamp" Text="Y" TStamp="Y"> </Prop>
        <Prop DBCol="AppDomainsLoaded"
ThreshDerived="Y">Currentappdomains</Prop>
        <Prop DBCol="AssembliesLoaded" ThreshDerived="Y">Current
Assemblies</Prop>
```

```
        <Prop DBCol="ClassesLoaded"
ThreshDerived="Y">CurrentClassesLoaded</Prop>
        <Prop DBCol="ModulesLoaded" ThreshDerived="Y"> </Prop>
        <Prop DBCol="AssemblyReloads" ThreshDerived="Y"> </Prop>
        <Prop DBCol="ModuleReloads" ThreshDerived="Y"> </Prop>
        <Prop DBCol="ClassReloads" ThreshDerived="Y"> </Prop>
        <Prop DBCol="JitPitches" ThreshDerived="Y"> </Prop>
        <Prop DBCol="ClassLoadFailures"
ThreshDerived="Y">TotalNumberofLoadFailures</Prop>
        <Prop DBCol="TotalClassesLoaded"
ThreshDerived="Y">TotalClassesLoaded</Prop>
        </DatabaseTbl>
```

Correlated portion

```
        <entity id="e6">
                <description>Processor</description>
                <item selected="1">Win32_Processor</item>
                <oncontextmenu> </oncontextmenu>
                <image>../../tndportal/images/map-service</image>
                <imageOpen>../../tndportal/images/map-
service_mgn.gif</imageOpen>
                <onContextMenu>context/contextAssembly.xx</onContextMenu>
                <contents>
                        <entity id="e61">
                        <description>AllProcessors</description>
                        <item selected="1"> </item>
                        <image>../..tndportal/images/map-fsys</image>
                <imageOpen>../../tndportal/images/map-fsys</imageOpen>
            <onContextMenu>context/contextClass.xxx</onContextMenu>
                    <contents>
                    <entity id="e611">
            <description>CPUClockSpeed</description>
                    <item selected="1">Current ClockSpeed</item>
                    <image>../../tndportal/images/map-files</image>
                            <imageOpen>../../tndportal/images/map-
    files</imageOpen>
                            <onClick> </onClick>
            <onContextMenu>context/contextFunction.xx</onContextMenu>
                    </entity>
                    <entity id="e612">
                    <description>CPULoad</description>
                    <item selected="1">LoadPercentage</item>
                    <image>../../tndportal/images/map-files</image>
            <imageOpen>../../tndportal/images/map-files</imageOpen>
                    <onClick> </onClick>
            <onContextMenu>context/contextFunction.xx</onContextMenu>
                    </entity>
                    </contents>
                </entity>
            </contents>
        </entity>
```

It will be understood that the above illustrations are for example purposes only and mapping table 170 may include none, some, or all of the illustrated tags, as well as other tags and data structures, without departing from the scope of the disclosure.

Server 102 also includes processor 125. Processor 125 executes instructions and manipulates data to perform the operations of server 102 such as, for example, a central processing unit (CPU), an application specific integrated circuit (ASIC) or a field-programmable gate array (FPGA). Although FIGURE 1 illustrates a single processor 125 in server 102, multiple processors 125 may be used according to particular needs, and reference to processor 125 is meant to include multiple processors 125 where applicable. In the embodiment illustrated, processor 125 executes management engine 115 that processes metrics 140 for use in system 100. Management engine 115 could include any hardware, software, firmware, or combination thereof operable to receive or retrieve metrics 140 from extensible instrumentation layer 130 (such as WMI), automatically map metrics 140 to one or more SQL statements 160 based on mapping table 170, and transform the data such that any data repository or display, such as database 124, may store or reference it. For example, management engine 115 may provide client 104 with data displays 150, reports 150, or management interfaces 150 operable to view and manipulate the stored metrics 140. It will be understood that while management engine 115 is illustrated as a single multi-tasked module, the features and functionality performed by this engine may be performed by multiple modules such as, for example, a retrieving module, a transforming module, and an editing module. Moreover, while not illustrated, management engine 115 may be a child or sub-module of any other appropriate software module such as, for example, an enterprise infrastructure management application without departing from the scope of this disclosure.

Server 102 also often includes interface 112 for communicating with other computer systems, such as client 104, over network 108 in a client-server or other distributed environment via link 118. In certain embodiments, server 102 receives metrics 140 from a plurality of distributed nodes 130, as illustrated in FIGURE 2B, via network 108 for storage in memory 120. Network 108 facilitates wireless or wireline communication between computer server 102 and any other computer.

Network 108 may communicate, for example, Internet Protocol (IP) packets, Frame Relay frames, Asynchronous Transfer Mode (ATM) cells, voice, video, data, and other suitable information between network addresses. Network 108 may include one or more local area networks (LANs), radio access networks (RANs), metropolitan area networks (MANs), wide area networks (WANs), all or a portion of the global computer network known as the Internet, and/or any other communication system or systems at one or more locations. Generally, interface 112 comprises logic encoded in software and/or hardware in a suitable combination and operable to communicate with network 108. More specifically, interface 112 may comprise software supporting one or more communications protocols associated with communications network 108 or hardware operable to communicate physical signals.

Client 104 comprises any computer and may include input devices, output devices, mass storage media, processors, memory, interfaces, communication ports, or other appropriate components for communicating formatted metrics 140 to the user of client 104. It will be understood that there may be any number of clients 104 coupled to server 102 or client 104 may comprise a management component of server 102. As used in this document, client 104 is intended to encompass a personal computer, workstation, network computer, kiosk, wireless data port, personal data assistant (PDA), one or more processors within these or other devices, or any other suitable processing or display device. Moreover, "client 104" and "user of client 104" may be used interchangeably without departing from the scope of this disclosure. For example, client 104 may comprise a computer that includes an input device, such as a keypad, touch screen, mouse, or other device that can accept information, and an output device that conveys information associated with the operation of server 102 or clients 104, including digital data, visual information, or metrics 140. Both the input device and output device may include fixed or removable storage media such as a magnetic computer disk, CD-ROM, or other suitable media to both receive input from and provide output to users of clients 104 through a portion of the web product interface, namely graphical user interface (GUI) 116.

GUI 116 comprises a graphical user interface operable to allow the user of client 104 to interface with system 100 and view the output of a plurality of software products. Generally, GUI 116 provides the user of client 104 with an efficient and user-friendly presentation of data provided by system 100, such as a display or report of one or more database tables storing metrics 140. GUI 116 may comprise a plurality of displays having interactive fields, pull-down lists, and buttons operated by the user. In one example, GUI 116 presents the formatted output and receives commands from client 104. It should be understood that the term graphical user interface may be used in the singular or in the plural to describe one or more graphical user interfaces and each of the displays of a particular graphical user interface. Further, GUI 116 contemplates any graphical user interface, such as a generic web browser, that processes information in system 100 and efficiently presents the information to the user. Server 102 can accept data from client 104 via the web browser (e.g., Microsoft Internet Explorer or Netscape Navigator) and return the appropriate HTML or eXtensible Markup Language (XML) responses. For example, GUI 116 may comprise a front-end of management engine 115. Accordingly, for ease of understanding, the term GUI 116 and management engine 115 may be used interchangeably; although, it will be understood that management engine 115 will often include more functionality than a graphical user interface.

In one aspect of operation, management engine 115 interfaces metrics 140 into database 124 based on mappings defined in one or more mapping tables 170. Generally, management engine 115 maps individual classes to database tables and properties within those classes to columns within the respective database tables. According to certain embodiments, the mapping of metrics 140 to database-compliant structures depends on whether metrics are .NET specific or correlated. More specifically, if metrics 140 are correlated, then management engine 115 at least partially maps based on an instance of each particular class and one of the mapping tables 170. Whereas if metrics 140 are related to or otherwise associated with the .NET framework, then management engine 115 may locate one or more tags and sub-tags in the appropriate mapping table 170 for data mapping between the classes and properties to tables and columns. Once management engine 115 determines the type of metrics 140 and identifies the appropriate data map, then engine 115 generates one or more SQL statements 160 based on the identified data map. Accordingly, class-based metrics 140 are then in table-based form and are easily readable and manipulated by the user or another process.

FIGURES 2A-B illustrate alternative configurations of scaleable system 100. At a high level, FIGURE 2A illustrates a local configuration of computer 202 operable to store metrics 140 in a database 124 or present metrics 140 to a user through GUI 116 in table format and FIGURE 2B illustrates a distributed architecture such as, for example, a portion of an enterprise. In FIGURE 2A, computer 202 includes three logical layers: instrumentation layer 132, data layer 122, and presentation layer 128. These three logical layers may be part of management engine 115 or may be communicably coupled with management engine 115 without departing from the scope of this disclosure. Instrumentation layer 132 communicates with the .NET framework through example Windows Management Instrumentation 130, which generates, populates, or formats standard metrics 140. Management engine 115 transforms, maps, or converts metrics 140 into database-compliant data structures, such as SQL script 160, and communicates the data structure to database 124. Using any suitable technique, these data structures are loaded or interfaced with database 124. The logical presentation layer is operable to present the information stored in example SQL script 160 to a user through GUI 116. For example, presentation layer 128 may present one or more HTML pages generated based on SQL script 160.

In FIGURE 2B, system 100 distributes portions of processing among a plurality of instrumentation nodes 130, server 102, and one or more client browsers 116. In this embodiment, instrumentation layer 132 is remote from and communicably coupled with server 102 and then segmented among the plurality of nodes 130. For example, each node 130 may be an extensible instrumentation agent operable to monitor a unique portion of a data processing environment for extensible instrumentation layer 130. Each instrumentation node 130 collects metric information and communicates metrics 140 to server 102 for processing. Once server 102 receives metrics 140, management engine 115 generates SQL script 160 for storage in database 124. Further, server 102 may generate graphical displays based on metrics 140, SQL scripts 160, or database 124. These graphical displays may then be presented, using presentation layer 126, to one or more clients through any number of GUIs 116.

FIGURE 3 is an example data flow diagram illustrating the mapping of metrics 140 to a database. In the illustrated embodiment, metrics 140 are identified as one of two example categories, .NET framework metrics 140a and other/correlated metrics 140b. Management engine 115 retrieves, receives, selects, or otherwise identifies metrics 140a and/or 140b for storage in database 124 or presentation to a particular user, such as a system administrator. For example, extensible instrumentation layer 130 may automatically communicate metrics 140a to management engine 115. In another example, the particular user may manually instruct management engine 115 to retrieve metrics 140b from extensible instrumentation layer 130. Once management engine 115 identifies metrics 140a or 140b, engine 115 parses, maps, converts, or processes metrics 140a and/or metrics 140b based, at least in part, on first or second mapping tables 170a and 170b. In one embodiment, first mapping table 170a may be a document operable to map from .NET framework metrics 140a to one or more SQL statements 160 and second mapping table 170b may be a document operable to map from correlated metrics 140a to one or more SQL statements 160. As described above, mapping tables 170a and 170b may each represent a portion of one document, array, or object without departing from the scope of the disclosure. At any appropriate time, management engine 115 generates the one or more SQL scripts 160 or files based on the determined data mappings. Then, management engine 115 (or some other component such as a DBMS) may load into or otherwise interface SQL scripts 160 with database 124 for addition, modification, or deletion of table schemas or data within database 124.

FIGURES 4A-B illustrate various example displays of database tables storing metrics 140. It will be understood that each view is for illustration purposes only and system 100 may utilize or present any graphical display in any format, with any suitable data, without departing from the scope of this disclosure. FIGURE 4A illustrates one view 402 of a subset of metrics 140 received from extensible instrumentation layer 130. In this example display, GUI 116 presents metrics 140 associated with CLR memory WMI class. As illustrated, the WMI class is presented in table format based on conversion into a database-compliant structure. This table is generated based on the CLR memory WMI class and the various columns (AllocatedBytes, FinalizationSurvivor, Hen0heapsize, Gen0Promoted, and Gen1heapsize, for example) are generated based on the attributes defined within the WMI class. Each row may be generated or populated based portions of interfaced metrics 140. For example, management engine 115 may receive similar metrics 140 from two different environments, aspnet_wp and devenv for example, and generate two rows based on these two environments. FIGURE 4B illustrates a second example view 404 of data utilized in system 100. In this example, view 404 presents various properties of a class from metrics 140 and descriptive information about each property such as, for example, name, type, and value. Management engine 115 uses this information or mapping table 170, which may include similar information, to map class and property information in metrics 140 to database-compliant data structures for use by database 124, as well as other components in system 100.

FIGURES 5A-B are flow diagrams illustrating example methods 500 and 550, respectively, for storing metrics 140 in database 124 according to various embodiments of the disclosure. Generally, FIGURE 5A illustrates method 500, which converts .NET framework metrics 140, and FIGURE 5B illustrates method 550, which converts correlated metrics 140. The following descriptions focus on the operation of management engine 115 in performing methods 500 and 550. But system 100 contemplates using any appropriate combination and arrangement of logical elements implementing some or all of the described functionality.

Method 500 begins when management engine 115 extracts, receives, or otherwise collects metrics 140 from extensible instrumentation layer 130 at step 502. In the illustrated embodiment, management engine 115 determines if extracted metrics 140 are .NET specific at decisional step 504. If metrics 140 are not .NET specific then method 500 ends and, perhaps, method 550 begins. Otherwise, management engine 115 retrieves a mapping table 170 associated with .NET metrics 140 at step 506. Once metrics 140 have been collected and the appropriate mapping table 170 has been located, processing proceeds to steps 508 through 526, where metrics 140 are mapped to one or more database-compliant data structures based on mapping table 170.

Management engine 115 begins mapping the extracted metrics 140 by selecting a first class from extracted metrics 140 at step 508. Next, management engine 115 locates a tag in mapping table 170 that is associated with the selected class at step 510. Once located, management engine 115 maps the selected class to a database table defined by the located tag. It will be understood that the database table in database 124 may be previously defined or undefined without departing from the scope of this disclosure. For example, the mapping of the selected class with the database table may comprise associating the class with a pre-defined database table. In another example, management engine 115 may create a database table using the name of the selected class, class level attributes, and other appropriate class characteristics. Next, at step 514, management engine 115 selects the first property in the selected class. Based on this selected property, management engine 115 locates a subtag within the current tag in mapping table 170 at step 516. At step 518, management engine 115 maps the selected property in the selected class to a column in the associated database table as defined by the subtag. As with step 512, this mapping may be to a previously defined or undefined column in the database table. For example, if the column was previously undefined, management engine may define the column attributes, such as data type and data length, based on the selected property and subtag. Next, at decisional step 520, management engine 115 determines if there are more properties in the selected class. If there are, then management engine 115 selects the next property in the selected class at step 522 and execution returns to step 516. Once there are no more properties in the selected class, processing proceeds to decisional step 524. At decisional step 524, management engine 115 determines if there are more classes in extracted metrics 140. If there are, management engine 115 selects the next class from extracted metrics 140 at step 526 and execution returns to step 510.

Once there are no more unmapped classes in extracted metrics 140 at decisional step 524, management engine 115 generates a SQL statement or script 160 at step 528 based on the mappings determined in steps 508 through 526. It will be understood that SQL statement 160 may be used to define a column or database table, upload data into database 124 based on metrics 140, update data already stored in database 124, or to perform any other appropriate database operation. Next, at step 530, management engine 115 may load or interface the generated SQL statement 160 into database 124.

Turning to FIGURE 5B, method 550 begins at step 552 when management engine 115 extracts, receives, or otherwise collects or identifies metrics 140 from extensible instrumentation layer 130 at step 552. In the illustrated embodiment, management engine 115 determines if extracted metrics 140 are correlated at decisional step 554. If metrics 140 are not correlated then method 550 ends. Otherwise, management engine 115 retrieves, selects, or identifies a mapping table 170 associated with extracted correlated metrics 140 at step 556. Once metrics 140 have been collected and the appropriate mapping table 170 has been located, processing proceeds to step 558 through 580, where metrics 140 are mapped to database 124 based on retrieved mapping table 170.

Management engine 115 begins mapping the extracted metrics 140 by selecting a first class from extracted metrics 140 at step 558. Next, management engine 115 locates a tag in mapping table 170 that is associated with the selected class at step 560. Once located, at step 562 management engine 115 identifies an instance of the selected class. Based on this instance, management engine 115 locates a subtag within the located tag in mapping table 170 at step 564. Management engine 115 then maps the selected class instance to a database table defined by the located subtag at step 566. As described in FIGURE 5A, it will be understood that the database table in database 124 may be previously defined or undefined without departing from the scope of this disclosure. Next, at step 568, management engine 115 selects the first property in the selected class instance. Based on this selected property, management engine 115 locates a second-tier (or child level) subtag within the current class instance subtag in mapping table 170 at step 570. At step 572, management engine 115 maps the property in the selected class instance to a column in the database table as defined by the second level subtag. As with step 564, this mapping may be to a previously defined or undefined column in the database table. For example, if the column was previously undefined, management engine may define the column attributes such as data type and data length based on the selected property. Next, at decisional step 574, management engine 115 determines if there are more properties in the selected class. If there are, then management engine 115 selects the next property in the selected class at step 576 and execution returns to step 570. Once there are no more properties in the selected class, processing proceeds to decisional step 578. At decisional step 578, management engine 115 determines if there are more classes in extracted metrics 140. If there are, management engine 115 selects the next class from extracted metrics 140 at step 580 and execution returns to step 560.

Once there are no more unmapped classes in extracted metrics 140 at decisional step 578, management engine 115 generates a SQL statement 160 at step 582 based on the mappings from step 558 through 580. It will be understood that SQL statement 160 may be used to define a column or database table, upload data into database 124 based on metrics 140, update data already stored in database 124 or any other appropriate database operation. Next, at step 584, management engine 115 may load or interface the generated SQL statement 160 into database 124.

The preceding flowcharts and accompanying description illustrate only exemplary methods 500 and 550. In short, system 100 contemplates using any suitable technique for performing these and other tasks. Accordingly, many of the steps in these flowcharts may take place simultaneously and/or in different orders than as shown. Moreover, system 100 may use methods with additional steps, fewer steps, and/or different steps, so long as the methods remain appropriate.

Although this disclosure has been described in terms of certain embodiments and generally associated methods, alterations and permutations of these embodiments and methods will be apparent to those skilled in the art. Accordingly, the above description of example embodiments does not define or constrain this disclosure. Other changes, substitutions, and alterations are also possible without departing from the spirit and scope of this disclosure.

WHAT IS CLAIMED IS:

1. A method for storing data associated with an extensible instrumentation layer comprises:

receiving metrics from an extensible instrumentation layer in an operating system, the metrics defined by at least one class and a plurality of properties, each property associated with one class;

converting the metrics into at least one database-compliant data structure; and

storing the one or more data structures in a database.

2. The method of Claim 1, the extensible instrumentation layer comprising Windows Management Instrumentation (WMI).

3. The method of Claim 1 further comprising:

mapping a first class to a first database table;

mapping a first property associated with the first class to a first column in the first database table mapped from the first class; and

wherein converting the metrics into at least one database-compliant data structure comprises converting the metrics into at least one database-compliant data structure based on the maps.

4. The method of Claim 3, the mapping of each class and property being based on an eXtensible Markup Language (XML) file, the XML file comprising at least one tag and a plurality of sub-tags and each tag associated with at least one attribute of the metrics.

5. The method of Claim 4, the metrics associated with a .NET framework.

6. The method of Claim 5, each tag comprising a map between one class and one database table and each subtag comprising a map between one property and one column.

7. The method of Claim 3, further comprising generating a structured query language (SQL) statement based, at least in part, on the mappings.

8. The method of Claim 3, the metrics defined by a plurality of classes and the method further comprising mapping a relationship between two of the plurality of classes and a relationship between two database tables.

9. The method of Claim 4, further comprising:

receiving a second set of metrics from the extensible instrumentation layer;

mapping the second set of metrics to at least one database-compliant data structure based on a second XML file;

converting the second of metrics into the one or more database-compliant data structures; and

storing the one or more data structures for the converted second set of metrics in the database.

10. The method of Claim 9, each set of metrics associated with a metric category and the method further comprising selected the first and second XML file based on each metric category.

11. The method of Claim 10, the first XML file and the second XML file comprising first and second portions of one XML document.

12. The method of Claim 1, each metric selected from the group consisting of:

load time for one or more assemblies, modules, and classes;

load count of one or more assemblies, modules, and classes;

load failures for one or more classes;

jit compile time for one or more functions;

jit search time for one or more functions;

jit count for one or more functions;

jit compile failures for one or more functions;

function execution time for one or more functions;

function exceptions for one or more functions; or function interop boundary crossings for one or more functions.

13. The method of Claim 1, wherein receiving metrics from the extensible instrumentation layer comprises polling the extensible instrumentation layer at a predetermined interval.

14. The method of Claim 4, the metrics comprising correlated metrics.

15. The method of Claim 14, further comprising:

determining one class instance for each class of the metrics using the XML file; and

mapping each class to one database table based on the determined class instance.

16. Software for storing data associated with an extensible instrumentation layer operable to:

receive metrics from an extensible instrumentation layer in an operating system, the metrics defined by at least one class and a plurality of properties, each property associated with one class;

convert the metrics into at least one database-compliant data structure; and

store the one or more data structures in a database.

17. The software of Claim 16, the extensible instrumentation layer comprising Windows Management Instrumentation (WMI).

18. The software of Claim 16 further operable to:

map one class to a database table;

map a first property associated with the one class to a first column in the database table mapped from the associated class; and

wherein the software operable to convert the metrics into at least one database-compliant data structure comprises software operable to convert the metrics into at least one database-compliant data structure based on the maps.

19. The software of Claim 18, the mapping of each class and property being based on an eXtensible Markup Language (XML) file, the XML file comprising

at least one tag and a plurality of sub-tags and each tag associated with at least one attribute.

20. The software of Claim 19, the metrics associated with a .NET framework.

21. The software of Claim 20, each tag comprising a map between one class identified by the attribute and one database table and each subtag comprising a map between one property and one column.

22. The software of Claim 18, further operable to generate a structured query language (SQL) statement based, at least in part, on the mappings.

23. The software of Claim 18, the metrics defined by a plurality of classes and the software further operable to map a relationship between two of the plurality of classes and a relationship between two database tables.

24. The software of Claim 19 further operable to:

receive a second set of metrics from the extensible instrumentation layer;

map the second set of metrics to at least one database-compliant data structures based on a second XML file;

convert the second of metrics into the one or more database-compliant data structures; and

store the one or more data structures for the converted second set of metrics in the database.

25. The software of Claim 24, each set of metrics associated with a metric category and the software further operable to identify the first and the second XML file based on each metric category.

26. The software of Claim 25, the first XML file and the second XML file comprising a first and second section of one XML document.

27. The software of Claim 16, each metric selected from the group consisting of:

load time for one or more assemblies, modules, and classes;

load count of one or more assemblies, modules, and classes;

load failures for one or more classes;

jit compile time for one or more functions;

jit search time for one or more functions;

jit count for one or more functions;

jit compile failures for one or more functions;

function execution time for one or more functions;

function exceptions for one or more functions; or

function interop boundary crossings for one or more functions.

28. The software of Claim 16, wherein the software operable to receive metrics from the extensible instrumentation layer comprises software operable to poll the extensible instrumentation layer at a predetermined interval.

29. The software of Claim 19, the metrics comprising correlated metrics.

30. The software of Claim 29, further operable to:

determine one class instance for each class of the metrics using the XML file; and

map each class to one database table based on the determined class instance.

31. A system for storing data associated with an extensible instrumentation layer comprising:

memory operable to store one or more mapping files and a database;

one or more processors operable to:

receive metrics from an extensible instrumentation layer in an operating system, the metrics defined by at least one class and a plurality of properties, each property associated with one class;

convert the metrics into at least one database-compliant data structure based on one of the mapping files; and

store the one or more data structures in the database.

32. The system of Claim 31, the extensible instrumentation layer comprising Windows Management Instrumentation (WMI).

33. The system of Claim 31, the one or more processors further operable to:

map one class to a database table based on the mapping file;

map a first property associated with the one class to a first column in the database table mapped from the associated class based on the mapping file; and

wherein the one or more processors operable to convert the metrics into at least one database-compliant data structure based on one of the mapping files comprises one or more processors operable to convert the metrics into at least one database-compliant data structure based on the maps.

34. The system of Claim 31, the mapping file comprising an eXtensible Markup Language (XML), the XML file comprising at least one tag and a plurality of sub-tags, each tag associated with at least one attribute.

35. The system of Claim 34, the metrics associated with a .NET framework.

36. The system of Claim 35, each tag comprising a map between one class identified by the attribute and one database table and each subtag comprising a map between one property and one column.

37. The system of Claim 33, the one or more processors further operable to generate a structured query language (SQL) statement based, at least in part, on the mappings.

38. The system of Claim 33, the metrics defined by a plurality of classes and the software further operable to map a relationship between two of the plurality of classes and a relationship between two database tables.

39. The system of Claim 31, the one or more processors further operable to:

receive a second set of metrics;

map the second set of metrics to at least one database-compliant data structures based on a second XML file;

convert the second of metrics into the one or more database-compliant data structures; and

store the converted second set of metrics in the database.

40. The system of Claim 39, each set of metrics associated with a metric category and the software further operable to select the first and second XML file based on each metric category.

41. The system of Claim 40, the first XML file and the second XML file comprising a first and second section of one XML document.

42. The system of Claim 31, each metric selected from the group consisting of:

load time for one or more assemblies, modules, and classes;

load count of one or more assemblies, modules, and classes;

load failures for one or more classes;

jit compile time for one or more functions;

jit search time for one or more functions;

jit count for one or more functions;

jit compile failures for one or more functions;

function execution time for one or more functions;

function exceptions for one or more functions; or

function interop boundary crossings for one or more functions.

43. The system of Claim 31, wherein the one or more processors operable to receive metrics from the extensible instrumentation layer comprises one or more processors operable to poll the extensible instrumentation layer at a predetermined interval.

44. The system of Claim 34, the metrics comprising correlated metrics.

45. The system of Claim 44, the one or more processors further operable to:

determine one class instance for each class of the metrics using the XML file; and

map each class to one database table based on the determined class instance.

46. A system for storing data associated with an extensible instrumentation layer comprises:

means for receiving metrics from an extensible instrumentation layer in an operating system, the metrics defined by at least one class and a plurality of properties, each property associated with one class;

means for converting the metrics into at least one database-compliant data structure; and

means for storing the one or more data structures in a database.

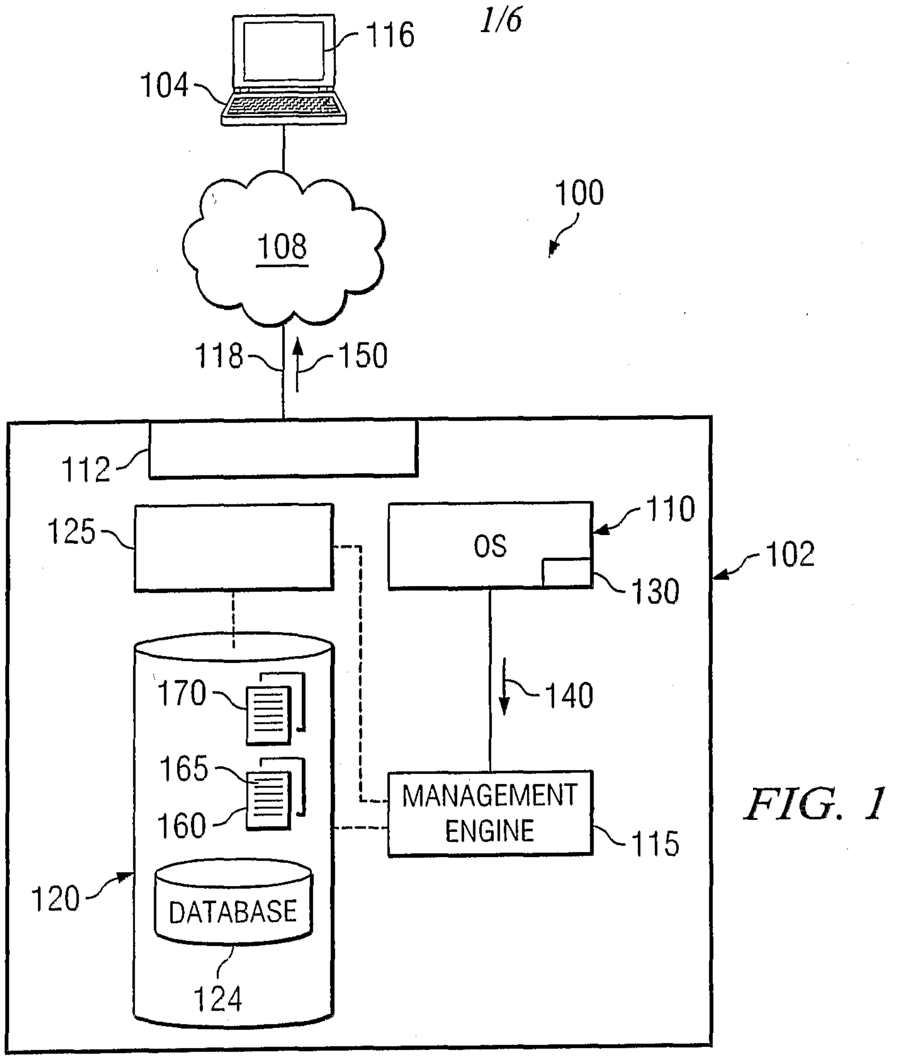

116
104
100
108
118
150
112
125
110
OS
130
102
140
170
165
160
MANAGEMENT ENGINE
115
120
DATABASE
124

FIG. 1

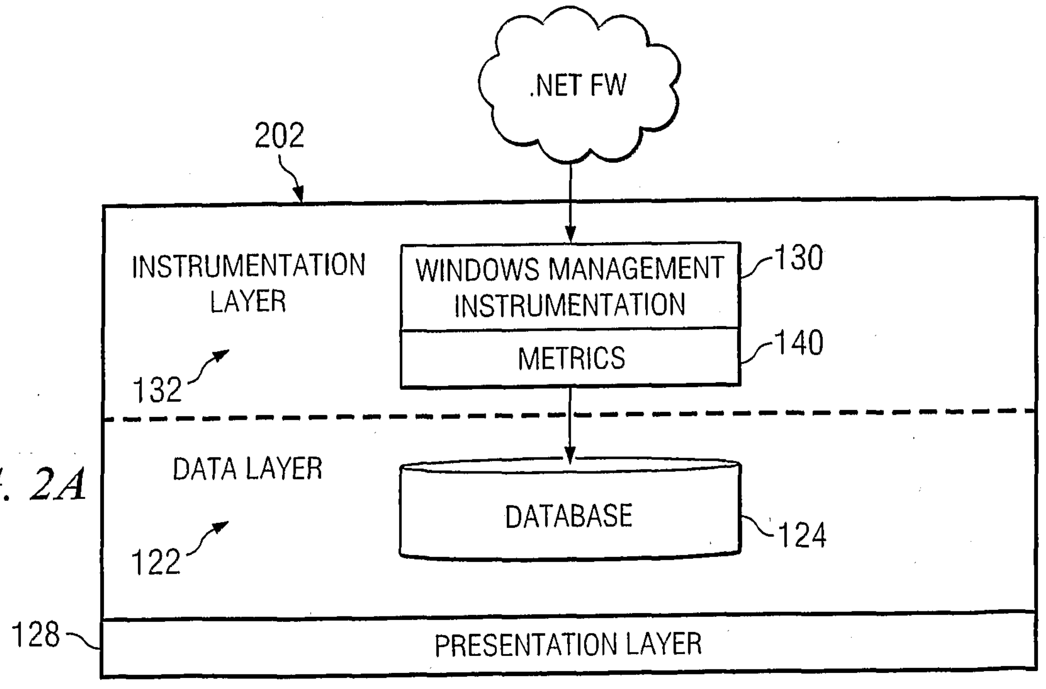

.NET FW
202
INSTRUMENTATION LAYER
132
WINDOWS MANAGEMENT INSTRUMENTATION
130
METRICS
140
DATA LAYER
122
DATABASE
124
128
PRESENTATION LAYER

FIG. 2A

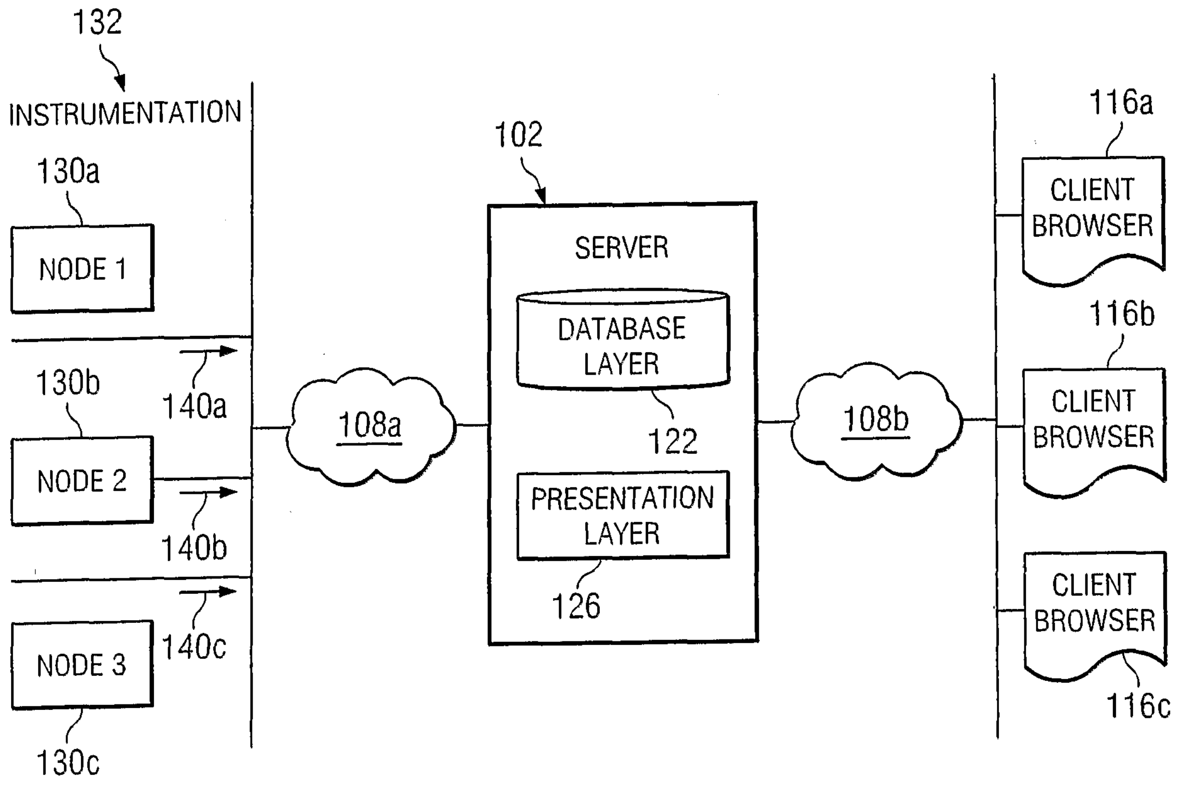

132
INSTRUMENTATION
130a
NODE 1
130b
NODE 2
NODE 3
130c
140a
140b
140c
108a
102
SERVER
DATABASE LAYER
122
PRESENTATION LAYER
126
108b
116a
CLIENT BROWSER
116b
CLIENT BROWSER
CLIENT BROWSER
116c

FIG. 2B

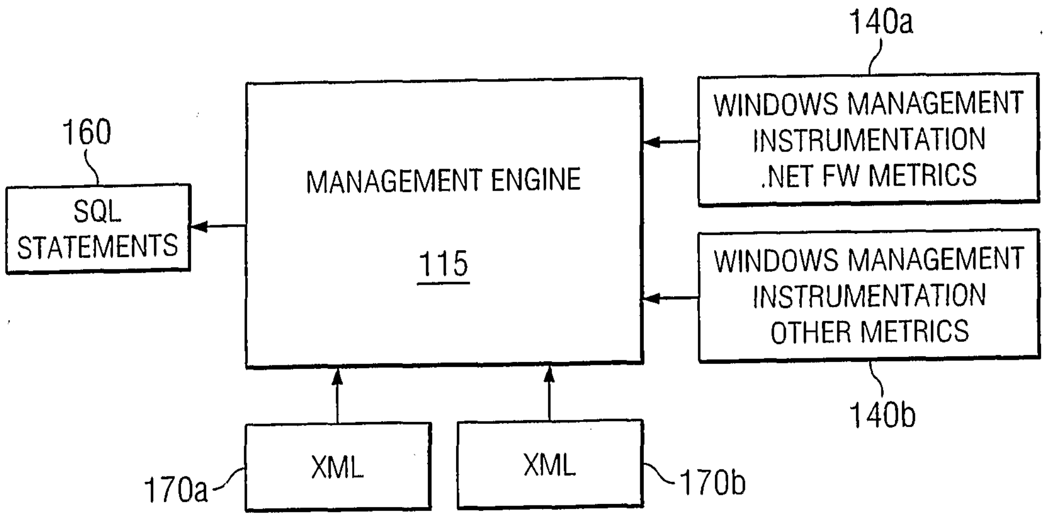
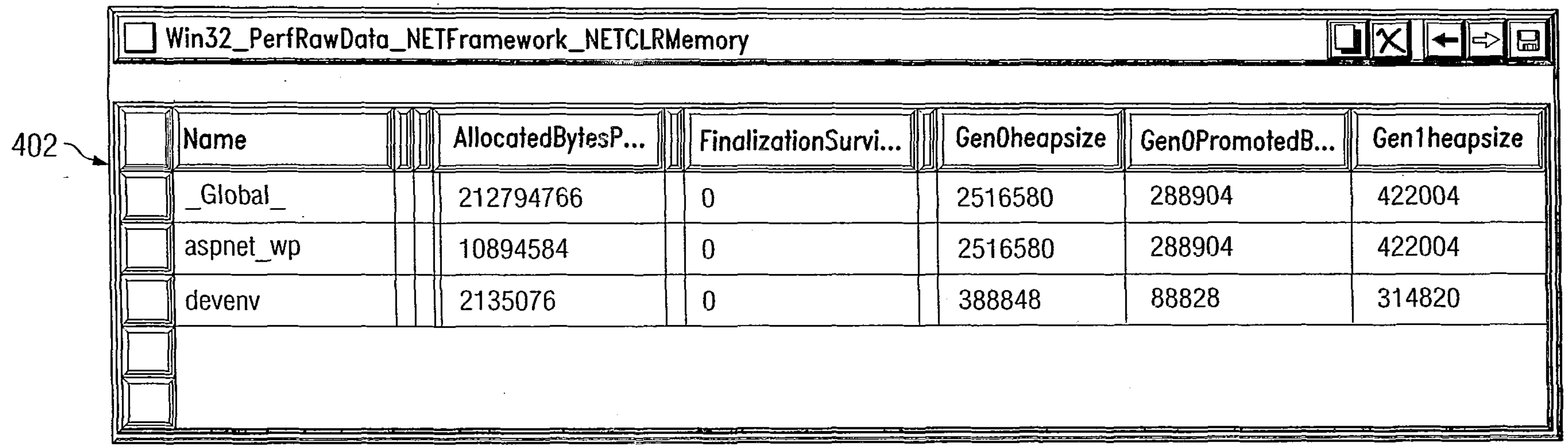
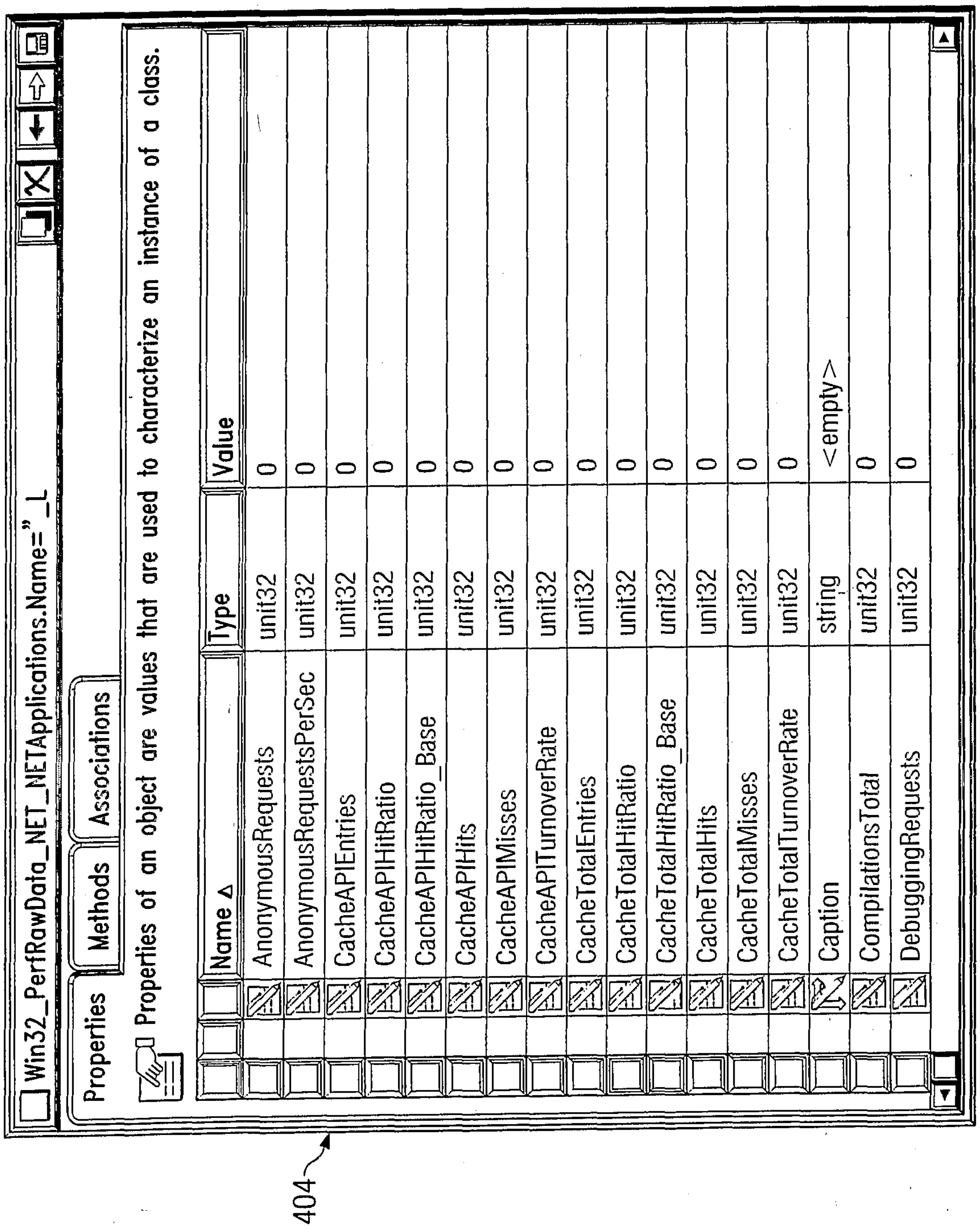

160
SQL STATEMENTS
MANAGEMENT ENGINE
115
140a
WINDOWS MANAGEMENT INSTRUMENTATION .NET FW METRICS
WINDOWS MANAGEMENT INSTRUMENTATION OTHER METRICS
140b
XML
XML
170a
170b

FIG. 3



Name	AllocatedBytesP...	FinalizationSurvi...	Gen0heapsize	Gen0PromotedB...	Gen1heapsize
Global	212794766	0	2516580	288904	422004
aspnet_wp	10894584	0	2516580	288904	422004
devenv	2135076	0	388848	88828	314820

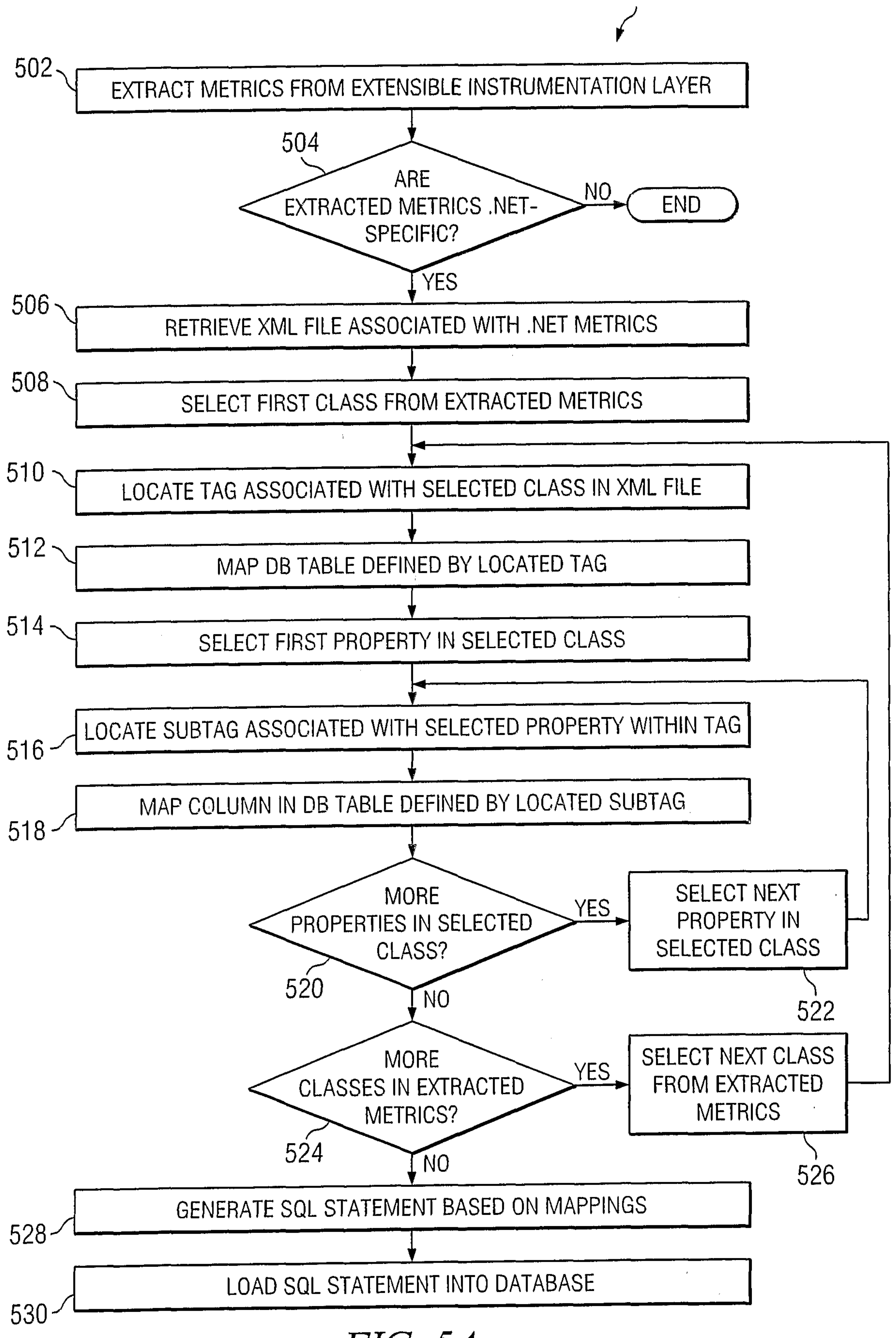
FIG. 4A



Properties of an object are values that are used to characterize an instance of a class.

Name ∆	Type	Value
AnonymousRequests	unit32	0
AnonymousRequestsPerSec	unit32	0
CacheAPIEntries	unit32	0
CacheAPIHitRatio	unit32	0
CacheAPIHitRatio_Base	unit32	0
CacheAPIHits	unit32	0
CacheAPIMisses	unit32	0
CacheAPITurnoverRate	unit32	0
CacheTotalEntries	unit32	0
CacheTotalHitRatio	unit32	0
CacheTotalHitRatio_Base	unit32	0
CacheTotalHits	unit32	0
CacheTotalMisses	unit32	0
CacheTotalTurnoverRate	unit32	0
Caption	string	<empty>
CompilationsTotal	unit32	0
DebuggingRequests	unit32	0

FIG. 4B



502
EXTRACT METRICS FROM EXTENSIBLE INSTRUMENTATION LAYER
504
ARE EXTRACTED METRICS .NET-SPECIFIC?
NO
END
YES
506
RETRIEVE XML FILE ASSOCIATED WITH .NET METRICS
508
SELECT FIRST CLASS FROM EXTRACTED METRICS
510
LOCATE TAG ASSOCIATED WITH SELECTED CLASS IN XML FILE
512
MAP DB TABLE DEFINED BY LOCATED TAG
514
SELECT FIRST PROPERTY IN SELECTED CLASS
516
LOCATE SUBTAG ASSOCIATED WITH SELECTED PROPERTY WITHIN TAG
518
MAP COLUMN IN DB TABLE DEFINED BY LOCATED SUBTAG
520
MORE PROPERTIES IN SELECTED CLASS?
YES
SELECT NEXT PROPERTY IN SELECTED CLASS
522
NO
524
MORE CLASSES IN EXTRACTED METRICS?
YES
SELECT NEXT CLASS FROM EXTRACTED METRICS
526
NO
528
GENERATE SQL STATEMENT BASED ON MAPPINGS
530
LOAD SQL STATEMENT INTO DATABASE

FIG. 5A

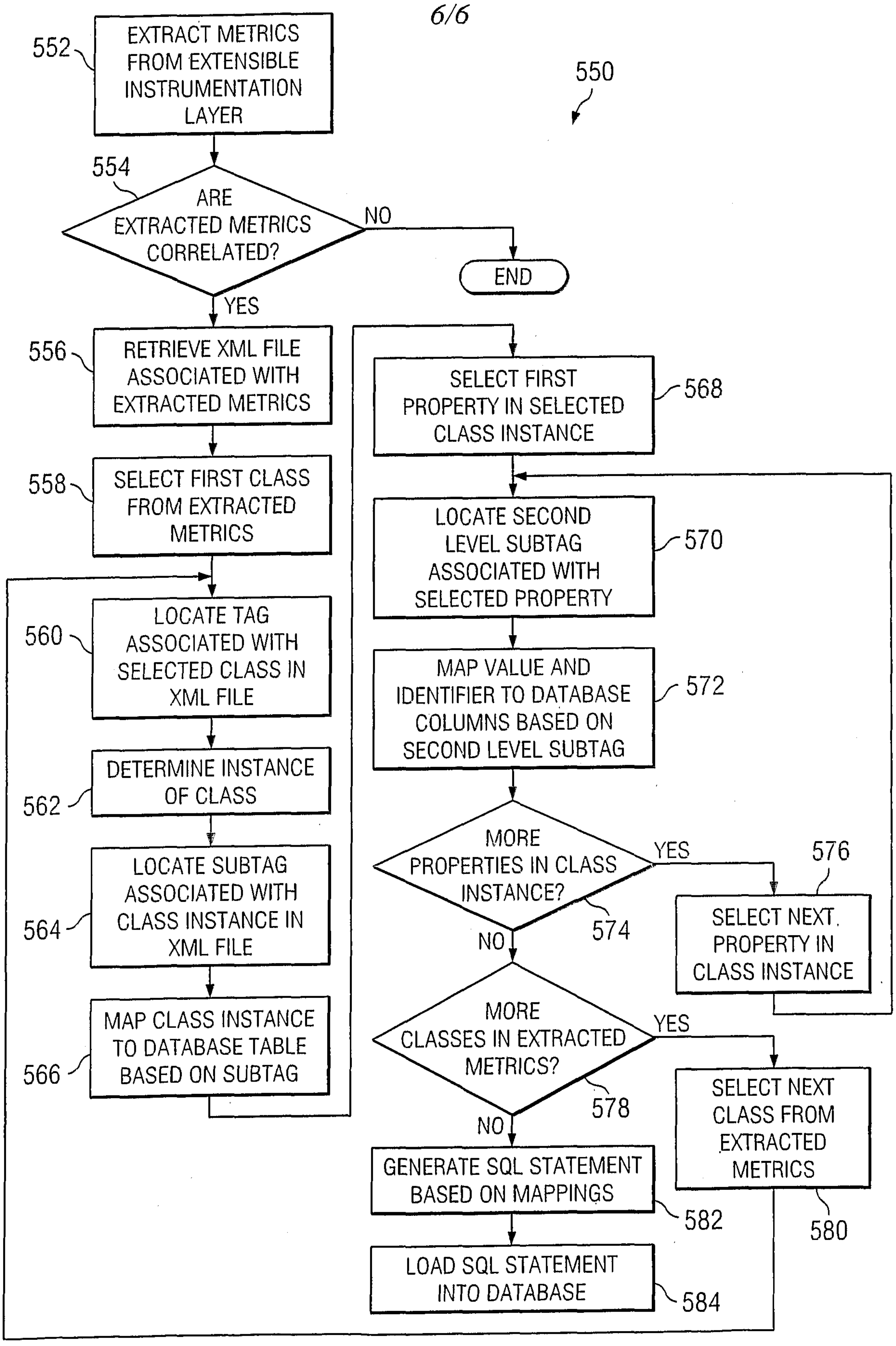

550
552
EXTRACT METRICS FROM EXTENSIBLE INSTRUMENTATION LAYER
554
ARE EXTRACTED METRICS CORRELATED?
NO
END
YES
556
RETRIEVE XML FILE ASSOCIATED WITH EXTRACTED METRICS
558
SELECT FIRST CLASS FROM EXTRACTED METRICS
560
LOCATE TAG ASSOCIATED WITH SELECTED CLASS IN XML FILE
562
DETERMINE INSTANCE OF CLASS
564
LOCATE SUBTAG ASSOCIATED WITH CLASS INSTANCE IN XML FILE
566
MAP CLASS INSTANCE TO DATABASE TABLE BASED ON SUBTAG
568
SELECT FIRST PROPERTY IN SELECTED CLASS INSTANCE
570
LOCATE SECOND LEVEL SUBTAG ASSOCIATED WITH SELECTED PROPERTY
572
MAP VALUE AND IDENTIFIER TO DATABASE COLUMNS BASED ON SECOND LEVEL SUBTAG
574
MORE PROPERTIES IN CLASS INSTANCE?
YES
576
SELECT NEXT PROPERTY IN CLASS INSTANCE
NO
578
MORE CLASSES IN EXTRACTED METRICS?
YES
580
SELECT NEXT CLASS FROM EXTRACTED METRICS
NO
582
GENERATE SQL STATEMENT BASED ON MAPPINGS
584
LOAD SQL STATEMENT INTO DATABASE

FIG. 5B

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US2004/022031

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	-/--	

[X] Further documents are listed in the continuation of box C. [ ] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search 29 October 2004	Date of mailing of the international search report 12/11/2004
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Authorized officer Eichenauer, L

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US2004/022031

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	"User Guide - Microsoft Operations Manager 2000" 'Online! 2001, MICROSOFT CORP , REDMOND, USA , XP002302452 Retrieved from the Internet: URL:http://www.microsoft.com/mom/docs/userg.pdf> 'retrieved on 2004-10-21! page 44, line 28 - page 45, line 2 page 18, line 1 - line 8	1-3,7,8, 12,13, 16-18, 22,23, 27,28, 31-33, 37-43,46
Y		4-6, 9-11,14, 15, 19-21, 24-26, 29,30, 34-36, 44,45
A	page 32, line 26 - line 29	13,28,43
X	CH. STEIGNER AND J. WILKE: "Multi-Source Performance Analysis of Distributed Software" PROCEEDINGS OF THE COMMUNICATION NETWORKS AND DISTRIBUTED SYSTEMS MODELING AND SIMULATION CONFERENCE, 'Online! 27 January 2002 (2002-01-27), XP002302448 SAN ANTONIO, TEXAS, USA Retrieved from the Internet: URL:http://www.uni-koblenz.de/{steigner/labor/papers/CNDS2002.pdf> 'retrieved on 2004-10-25! page 3 - page 6	1-3,7;8, 12,13, 16-18, 22,23, 27,28, 31-33, 37-43,46
X	"The Microsoft Windows Management Instrumentation - Extensions to the Windows Driver Model" MSDN LIBRARY, 'Online! September 1998 (1998-09), XP002302449 REDMOND, USA Retrieved from the Internet: URL:http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwmi/html/wmixwdm.asp> 'retrieved on 2004-10-25! page 1, line 18 - line 21; figure 1	1-3,7,8, 12,13, 16-18, 22,23, 27,28, 31-33, 37-43,46
Y		4-6, 9-11,14, 15, 19-21, 24-26, 29,30, 34-36, 44,45

-/--

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US2004/022031

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	"The Common Information Model - CIM Version 2.7" DISTRIBUTED MANAGEMENT TASK FORCE - TECHNICAL NOTE, 'Online! January 2003 (2003-01), XP002302450 PORTLAND, OREGON, USA Retrieved from the Internet: URL:http://www.dmtf.org/education/technote_CIM.pdf> 'retrieved on 2004-10-25! the whole document	1,16,31, 46
A	"Specification for the Representation of CIM in XML" 'Online! 2 June 1999 (1999-06-02), DISTRIBUTED MANAGEMENT TASK FORCE, INC (DMTF) , PORTLAND, OREGON, USA , XP002302453 Retrieved from the Internet: URL:http://www.dmtf.org/standards/documents/WBEM/CIM_XML_Mapping20.html> 'retrieved on 2004-10-25! the whole document	4-6, 9-11,14, 15, 19-21, 24-26, 29,30, 34-36, 44,45
A	"Common Information Model - Specification Version 2.2" 'Online! 14 June 1999 (1999-06-14), DISTRIBUTED MANAGEMENT TASK FORCE, INC. (DMTF) , PORTLAND, OREGON, USA , XP002302454 Retrieved from the Internet: URL:http://www.dmtf.org/standards/documents/CIM/DSP0004.pdf> 'retrieved on 2004-10-25! the whole document	1-46
A	JIM DAVIS: "WBEM Services Specification JSR-0048" JAVA DEVELOPER CONFERENCE, 'Online! 6 June 2001 (2001-06-06), XP002302451 SAN FRANCISCO, USA Retrieved from the Internet: URL:http://www.wbemsolutions.com/articles/jsr_48.pdf> 'retrieved on 2004-10-25! pages 17,21 pages 32-40	12,27,42