



(51) International Patent Classification:
G06F 11/34 (2006.01)

(21) International Application Number:
PCT/CN2017/078914

(22) International Filing Date:
31 March 2017 (31.03.2017)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, Redwood Shores, CA 94065 (US).

(72) Inventor; and

(71) Applicant (for MG only): **ZHANG, Weixuan** [CN/CN]; No. 24 Zhongguancun Software Park, Haidian, Beijing 100193 (CN).

(72) Inventors: **SHEN, Hui**; No. 24 Zhongguancun Software Park, Haidian, Beijing 100193 (CN). **TANG, Fangzhi**; No. 24 Zhongguancun Software Park, Haidian, Beijing 100193 (CN). **ZHU, Weiguo**; No. 24 Zhongguancun Software Park, Haidian, Beijing 100193 (CN).

(74) Agent: **CCPIT PATENT AND TRADEMARK LAW OFFICE**; 8th Floor, Vantone New World Plaza, 2 Fuchengmenwai Street, Xicheng District, Beijing 100037 (CN).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC,

(54) Title: SYSTEM AND METHOD FOR DETERMINING THE SUCCESS OF A CROSS-PLATFORM APPLICATION MIGRATION

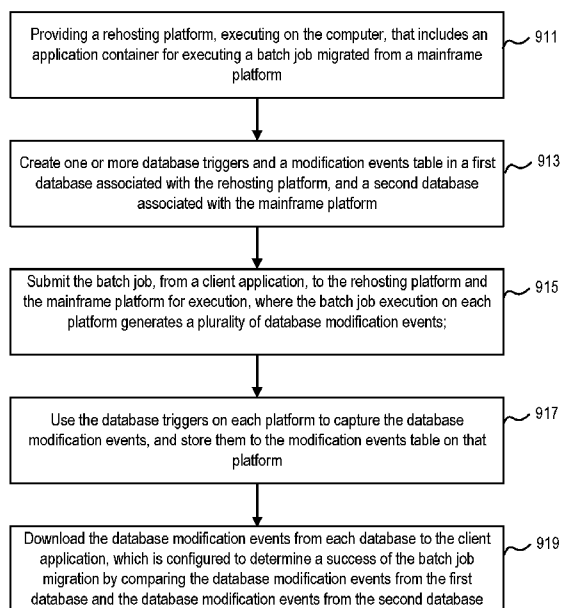


Figure 9

(57) Abstract: In accordance with an embodiment, described herein is a system and method for determining the migration success of an application from a mainframe platform to an open platform. A first database associated with the mainframe platform and a second database associated with the open platform can include the same data baseline. A set of triggers can be created on the first database and the second database, to capture database modification events generated by the execution of a job associated with the application on each platform, and to store the database modification events in a table in each database. The database modification events from the mainframe platform and the open platform can be downloaded and compared to determine the success of the application migration.

SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR,
TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

- (84) Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— with international search report (Art. 21(3))

SYSTEM AND METHOD FOR DETERMINING THE SUCCESS OF A CROSS-PLATFORM APPLICATION MIGRATION

COPYRIGHT NOTICE

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

Field of Invention:

[0001] Embodiments of the invention are generally related to application servers and application rehosting, and are particularly related to a system and method for determining the success of an application migration from a mainframe platform to an open platform.

Background:

[0002] To reduce costs and become as agile as possible, organizations today are increasingly seeking to move business-critical mainframe applications to open systems and cloud environments. However, to do so can often imply complex, costly and resource-heavy application migration projects, which deter companies from undertaking such migrations. As a result, organizations need a scalable, reliable and high-performance application migration solution that can help them transition to open systems and the cloud environments.

[0003] When a user migrates an mainframe application to an open system, e.g., Unix/Linux platform, a reliable method for determining the success of the migration is needed.

Summary:

[0004] In accordance with an embodiment, described herein is a system and method for determining the migration success of an application from a mainframe platform to an open platform. A first database associated with the mainframe platform and a second database associated with the open platform can include the same data baseline. A set of triggers can be created on the first database and the second database, to capture database modification events generated by the execution of a job associated with the application on each platform, and to store the database modification events in a table in each database. The database modification

events from the mainframe platform and the open platform can be downloaded and compared to determine the success of the application migration.

Brief Description of the Figures:

[0005] **Figure 1** illustrates an exemplary open system, in accordance with an embodiment.

[0006] **Figures 2-3** illustrate an example of the transactional middleware system as shown in Figure 1, in accordance with an embodiment.

[0007] **Figure 4** illustrates a system for providing a Job Control Language (JCL) execution engine in an open platform, in accordance with embodiment.

[0008] **Figure 5** illustrates a system for determining the success of an application migration from a mainframe platform to an open platform, in accordance with embodiment.

[0009] **Figure 6** further illustrates a system for determining the success of an application migration from a mainframe platform to an open platform, in accordance with embodiment.

[00010] **Figure 7** further illustrates a system for determining the success of an application migration from a mainframe platform to an open platform, in accordance with embodiment.

[00011] **Figure 8** illustrates a flowchart for determining the success of an application migration from a mainframe platform to an open platform, in accordance with embodiment.

[00012] **Figure 9** illustrates a method for determining the success of an application migration from a mainframe platform to an open platform, in accordance with embodiment.

Detailed Description:

[00013] As companies face increasing pressure to deliver more business value from their IT spending and free up funding for new business initiatives, more and more companies are migrating their mainframe applications to open systems/rehosting platforms to reduce mainframe costs and to modernize legacy applications.

[00014] As used in the document, a rehosting platform and an open platform can be used interchangeably. The OracleTM Tuxedo Application Runtimes (ART) is an example open platform or rehosting platform.

Rehosting Platform/Open Platform

[00015] **Figure 1** illustrates an exemplary open platform, in accordance with an embodiment.

[00016] As shown in Figure 1, a rehosting platform 101 and a rehosting workbench 125 can provide an open system/platform for rehosting mainframe applications and data 129 on

lower-cost platforms without losing business value or sacrificing Quality of Service (QoS).

[00017] In accordance with an embodiment, the mainframe applications and data 129 to be rehosted can currently run on a mainframe platform (mainframe platform) 127, for example, an IBM® mainframe platform; and can include a customer information control system (CICS) 131, an information management system (IMS) 135, a DB2 database 137, one or more data files (e.g., sequential files) 138, and a virtual storage access method (VSAM) file management system 139.

[00018] In accordance with an embodiment, the CICS and the IMS can be middleware products on the mainframe platform. The CICS is a heavy and rigid transaction processing management system designed to support rapid, high-volume online transaction processing. The IMS is a light-weight message-based transaction processing management system. The middleware products can be used to host business logic written in COBOL, PL/I, C, Assembly or 4GLs. The VSAM can comprise different file organizations which can be used by application programs to manage their data. The file organizations can include key sequenced data set key (KSDS), relative record data set (RRDS), entry sequenced data set (ESDS), and linear data set (LDS).

[00019] In addition, the mainframe platform 127 can include a batch execution environment 140 that can support JOB Control Language (JCL) 141 and a job entry subsystem (JES) 143. JCL can be a script language to implement batch processes on the mainframe platform. JES can be a major component of an operating system on the mainframe platform, can receive jobs into the mainframe platform, schedule the jobs for processing, and control their output processing.

[00020] As further shown in Figure 1, the rehosting platform can include a software stack compatible with the mainframe platform to run mainframe applications with little to no change to minimize the risks and cost of migration; and an integrated management and monitoring component 113 for use in monitoring the rehosting platform. The software stack can provide a set of mainframe-compatible functionalities to preserve CICS, IMS, and batch application logic and data.

[00021] In accordance with an embodiment, the software stack can include a plurality of application runtimes (ART) 109 for hosting mainframe applications, for example, a CICS application runtime 115, an IMS application runtime 117, and a batch application runtime 119. The plurality of application runtimes and a rehosting workbench 125 can be used to migrate 142 the mainframe applications 129 from the mainframe platform 127 to the rehosting platform 101.

[00022] In accordance with an embodiment, the CICS application runtime can include a

set of Tuxedo servers to simulate core features of the mainframe CICS. The Tuxedo system servers can provide underlying application server functions, including cluster management, request routing, health monitoring, restarts, failover, load balancing, transaction management, communication channels and gateways (ATMI, CICS, IMS, SOAP/HTTP web services, Java/JCA, .Net, ESB), and protocol conversion.

[00023] In accordance with an embodiment, the IMS application runtime can provide a set of DL/I calls for use by COBOL/C applications migrated from the mainframe platform 127; a robust session environment to handle concurrent connections from a plurality of 3270 terminals; a robust execution environment to provide OLTP to process transaction codes received from the 3270 terminals via calling the migrated COBOL/C applications; and a DB plug-in on the rehosting platform.

[00024] In accordance with an embodiment, the batch application runtime 119 can include a set of Tuxedo servers to simulate mainframe JES core features. For example, the batch application runtime can provide batch management and a plurality of JES functions (e.g., job queues, classes, priorities, and initiators).

[00025] In accordance with an embodiment, the rehosting workbench can be used to automate code and data migration using migration tools in the rehosting workbench. The code and data can include COBOL programs, copybooks, BMS screens, JCL, and DB2 DDL. The code and data can be transferred from the mainframe platform 127 to the rehosting workbench, which can parse source objects, calculate dependencies, generate metadata, and produce reports to indicate any missing objects or unused ones in the code and data.

[00026] In accordance with an embodiment, after the code and data are parsed, data migration tools for files and DB2 tables can run, followed by code migration tools for COBOL JCL. The code migration tools can apply sophisticated language processing to adapt COBOL code between compiler dialects, transform JCL, adapt SQL calls for differences between DB2 and Oracle DB. For data migration, the data migration tools can generate target schemas, including Oracle DDL, in the rehosting platform 101, and can automate data reloading to the generated target schemas.

[00027] In accordance with an embodiment, the rehosting workbench can be used in UNIX command line mode, and an Eclipse IDE graphical environment; and can generate system configuration files for the rehosting platform to facilitate configuration management and to simplify the deployment process.

[00028] The software stack can execute on a distributed transactional processing middleware system 121, for example, Oracle Tuxedo. The distributed transactional processing

middleware system can run on an open system environment, for example, UNIX, Linux, or Windows. The distributed transactional processing middleware system can include a native distributed architecture to provide transaction manager features for IMS and CICS from the perspective of applications.

[00029] In accordance with an embodiment, the distributed transactional processing middleware system can represent a transaction-oriented middleware, or an enterprise application server designed for high availability and to provide scalable applications to support transactions on various distributed systems.

[00030] Examples of the distributed transactional processing middleware system can include Tuxedo (Transactions for UNIX, Enhanced for Distributed Operation), a message-based communications system to distribute applications across various operating system platforms and databases.

[00031] Tuxedo allows messages to be queued to persistent or non-persistent storage (memory) for later processing or retrieval. An application-to-transaction monitor interface (ATMI) in Tuxedo can provide an interface that allows messages to be added to or read from queues. Tuxedo can pass service request messages between ATMI clients and servers through operating system (OS) inter-processes. In Tuxedo, requests are sent to named services, and Tuxedo uses memory based inter-process communication facilities to queue the requests to servers.

[00032] Rehosed mainframe applications can run as Tuxedo services, and can take advantage of SOA integration and enablement capabilities via a plurality of adapters 102, for example, a web service gateway adapter 103, an enterprise service bus (ESB) adapter 105, and an enterprise repository adapter 107.

[00033] In accordance with an embodiment, rehosed/migrated applications can be configured to expose a plurality of service interfaces in legacy components via standard WSDLs, and to provide robust bi-directional web services gateway capabilities. For example, the web service gateway adapter 103, an example of which can be Oracle Service Architecture Leveraging Tuxedo (SALT) adapter, can enable the rehosed applications to participate in SOA environments.

[00034] In addition, the rehosed applications can also use the ESB adapter 105 with built-in Tuxedo Transport for heterogeneous messaging. Web services and ESB end points can be imported into an enterprise repository via the enterprise repository adapter 107 to provide visibility, service dependency tracking, and other benefits of service lifecycle governance.

[00035] As further shown in Figure 1, in accordance with an embodiment, the distributed

transactional processing middleware system can execute on an engineered system and hardware 100, such as Oracle Exalogic and Oracle Exadata; and can include a clustered database 123, such as Oracle REAL Application Clusters. The clustered database can support usage of multiple individual systems as one clustered, virtual database server; and can provide transparent synchronization of read and write accesses to databases shared by all nodes in a cluster, dynamic distribution of database workload, and transparent protection against systems failures.

[00036] In accordance with an embodiment, the system described above, by combining a distributed transactional processing middleware system, a clustered database, an engineered system, and a plurality of open system products, can provide required reliability, availability, scalability and performance to rehosted mainframe applications.

[00037] **Figures 2-3** illustrate an example of the transactional middleware system 121 as shown in Figure 1, in accordance with an embodiment.

[00038] In accordance with an embodiment, Tuxedo (Transactions for UNIX, Enhanced for Distributed Operation) can be an example of the transactional middleware system. Tuxedo represents a middleware product or system that can operate as an extension of an operation system (for example, UNIX). As a platform for execution and development, Tuxedo is designed for the creation and administration of e-commerce online transaction processing (OLTP) systems.

[00039] In accordance with an embodiment, the Tuxedo system shown in Figures 2-3 can include an Application-to-Transaction Monitor Interface (ATMI) environment for use in communications, transactions, and management of data buffers.

[00040] As shown in Figure 2, the ATMI environment can include an external interface layer 211, an ATMI layer 213, a system services layer 215, and a resource manager layer 217. The external interface layer can provide a plurality of interfaces between users and the ATMI environment. The ATMI layer can represent an interface between applications and the ATMI environment. The system services layer can provide services and/or capabilities for developing and administering applications.

[00041] In accordance with an embodiment, the Tuxedo system can use a message-based communications system to distribute applications across various operating system platforms and databases.

[00042] As shown in Figure 3, communication within the ATMI environment can be accomplished by transferring messages. The Tuxedo system can pass service request messages between ATMI clients and servers through operating system (OS) inter-process

communications (IPC) message queues, for example, a request queue 311 and a reply queue 313. System messages and data can be passed between the OS-supported, memory-based queues of clients and servers in buffers.

[00043] In accordance with an embodiment, messages in the ATMI environment can be packaged in typed buffers, which can represent buffers that contain both message data and data identifying the types of message data being sent.

[00044] **Figure 4** illustrates a system for providing a Job Control Language (JCL) execution engine in an open platform, in accordance with embodiment.

[00045] In accordance with an embodiment, the batch application runtime 119 can provide a combination of utilities and services that allow a mainframe batch application to run unchanged, preserving existing business logic and data.

[00046] In accordance with an embodiment, a mainframe batch job can represent a unit of work that can comprise a plurality of job steps, and each job step can be specified by a set of job control statements in a JCL script. The JCL script can identify one or more programs to be executed, the inputs that are required, and location of the input/output.

[00047] As such, artifacts for a mainframe batch job can include a JCL script, and related programs (e.g., COBOL program) and data. The batch job can be submitted for execution using the JCL script via a client application.

[00048] In accordance with an embodiment, after a mainframe batch job is migrated from the mainframe platform to the rehosting platform, the JCL script of the mainframe batch job needs to be converted into one or more KSH scripts for execution in the rehosting platform.

[00049] Once converted, the mainframe application would be a different application, which makes the knowledge and skills of maintaining the original application useless, and makes it difficult to compare the converted application with the original application when the need arises to determine whether the converted application has executed successfully on the rehosting platform.

[00050] As shown in Figure 4, the rehosting platform 101 can provide native support for JCL jobs. The native support can enable the rehosting platform to run the JCL jobs without converting them.

[00051] In accordance with an embodiment, the batch application runtime 119 can provide a plurality of software modules/components in multiple layers to enable native support for JCL jobs, for example, a life cycle control layer 421, an execution layer 424, and a utilities/applications layer 431.

[00052] In accordance with an embodiment, the life cycle control layer can be configured

to manage the life cycle of a JCL job, and to manage high availability, scalability, management and monitoring of the JCL job. The execution layer can include a KSH execution engine 427 configured to parse a KSH job converted from a mainframe JCL job, and a JCL execution engine 429 configured to execute mainframe JCL jobs without converting them.

[00053] As shown in Figure 4, the life cycle control layer can include a logical job queue 411 that can receive submissions of KSH jobs 437 and JCL jobs 439 from a client application 409, and a plurality of job initiators (e.g., job initiator 413) distributed on a plurality of servers. The client application can be a command interface used to submit jobs, hold job, release jobs, cancel jobs, purge a jobs, display job information, or subscribe to events for job status changes. An example client application can be the Oracle Tuxedo client.

[00054] In accordance with an embodiment, the logical job queue can queue the KSH jobs and the JCL jobs, with each job associated with metadata specifying a job priority. The plurality of job initiators can be configured to dequeue jobs from the logical job queue, and dispatch each of the dequeued jobs, either to the job execution engine or the KSH execution engine, based on the nature of the job, for example, whether the job is a KSH job or a JCL job. There can be multiple running instances of the JCL execution engine, and multiple running instances of the KSH execution engine.

[00055] In accordance with an embodiment, the utilities/applications layer 431 can include one or more z/OS built-in utilities 433 and one or more applications 435 configured to be invoked by the JCL execution engine. The applications can be the programs identified for execution in a JCL script. An example application in this layer can be a COBOL program or a COBOL executable.

[00056] In accordance with an embodiment, when the JCL execution engine receives a JCL job, it can generate a statement sequence from the JCL job, and order statements in the sequence into a double-link sequence. The JCL execution engine can execute JCLLIB & INCLUDE statements in the JCL job to include code referenced by the JCL job, and parse the statement sequence to composite a logical job tree or a job element hierarchy. The job element hierarchy can include a plurality of steps (e.g., IF/THEN/ELSE/ENDIF, CNTL/ENDCNTL), each step containing one or more DDs, each DD containing one or more concatenated DDs. Finally, the JCL job can be executed.

DETERMINING MIGRATION SUCCESS

[00057] After a mainframe application is migrated to an open platform from a mainframe platform, a user would need to determine whether the migration is successful.

[00058] Generally, the migration success of a mainframe application can be determined by the execution success of a job that utilizes the programs, data and scripts associated with the application.

[00059] For example, the job's return code can be checked to determine if an expected value has been received. A zero means that the job has executed successfully, and a non-zero value means that a warning/error occurs. This type of verification can be checked from the exit code of a job execution process, or a job execution log file.

[00060] However, a job execution may require external resources, for example, data residing on either disk files, or database tables. As such, in addition to the job's return code, changes on external resources are also needed to be verified to determine the job has been executed successfully.

[00061] To verify data changes, a job needs to be launched on both the mainframe platform and the open platform, and database results from both platforms need to be compared to determine whether the job has performed the same data update operations on both platforms.

[00062] However, assuming the data baseline is the same for the mainframe platform and the open platform, directly comparing the results can be difficult, because of the size of the data, and the different database table definitions on the platforms.

[00063] In accordance with an embodiment, the system and method described herein can be used to determine the migration success of an application from a mainframe platform to an open platform. A first database associated with the mainframe platform and a second database associated with the open platform can include the same data baseline. A set of triggers can be created on the first database and the second database, to capture database modification events generated by the execution of a job associated with the application on each platform, and to store the database modification events in a table in each database. The database modification events from the mainframe platform and the open platform can be downloaded and compared to determine the success of the application migration.

[00064] In accordance with an embodiment, the first database can be a DB2 database, and the second database can be an Oracle database. The database triggers can be used to capture all data update operations resulted from the execution of the job on both platforms.

[00065] In accordance with an embodiment, a testing application can be configured to access both platforms to install database triggers on the DB2 and Oracle databases, and launch JCL jobs on the mainframe platform and the open platform.

[00066] For the DB2, a database trigger can be installed on each of a plurality of tables that the JCL job is associated with. For the Oracle database, database triggers can be installed

in a similar fashion.

[00067] In accordance with an embodiment, when the JCL job is launched from the testing application and executed on each platform, the database triggers on the database for each platform can be triggered to generate modification events. The testing application can be used to download the database modification events from each database, and invoke a comparison logic to compare the two sets of database modification events.

[00068] **Figure 5** illustrates a system for determining the success of an application migration from a mainframe platform to an open platform, in accordance with embodiment.

[00069] In accordance with an embodiment, mainframe application A 511 (a batch application) can be associated with a JCL script, one or more programs, and data; and can be migrated to the rehosting platform/open platform, as shown by application A' 513.

[00070] As shown by Figure 5, mainframe application A can be associated with a plurality of tables (for example, table A 529 and Table B 530) for storing data associated with the mainframe application in a mainframe database (e.g., a DB2 database) 537.

[00071] Similarly, application A' can be associated with a plurality of tables (for example, table A' 522 and Table B' 524) for storing data associated with the mainframe application in a rehosting database (e.g., an Oracle database) 521.

[00072] In accordance with an embodiment, a rehosting workbench can be used to recreate the plurality of tables associated with mainframe application A in the rehosting database, and clone the data in the plurality of tables to the rehosting platform, so that that the two databases can have the same data baseline.

[00073] As further shown in Figure 5, a test manager application 520, for example, Oracle test manager, can be used by a user 501 to create scripts for use in creating database triggers 523 for the plurality of tables in the rehosting database, and for creating database triggers 535 for the plurality of tables in the mainframe database. A modification events table (e.g., modification events table C 531 and modification events table C' 525) can be created respectively the mainframe database and the rehosting database, for storing database modification events captured by the database triggers in each platform.

[00074] In accordance with an embodiment, one or more batch jobs can be launched or submitted from the test manager application. As described above, the rehosting platform can provide a JCL execution engine for executing JCL jobs without the JCL jobs being converted them into KSH jobs. As such, after related programs and data are migrated from the mainframe platform to the rehosting platform, the JCL script of the batch application can be used to submit or launch batch jobs from the test manager application to both the rehosting platform and the

mainframe platform.

[00075] In an example implementation, the JCL script can reside in the test manager application, and can be submitted via FTP to each platform, to launch batch jobs on each platform.

[00076] In accordance with an embodiment, the database triggers in each database can listen on SQL statements issued by the batch application in each platform, capture database modification events generated by the SQL statements, and save the database modification events into a corresponding modification events table 525 or 531.

[00077] In accordance with an embodiment, when recording database modification events using the triggers on both platforms, the following rules apply: all NEW values are recorded for an insert operation; all OLD and NEW values are recorded for an UPDATE operation; and all OLD values are recorded for a DELETE operation.

[00078] In accordance with an embodiment, the test manager application can include a modification events comparison logic component 545. After execution of one or more batch jobs are completed on both platforms, database modification events data can be downloaded 547, 549 from the rehosting database and the mainframe database, and provided to the modification events comparison logic component. The modification events comparison logic component can compare the modification events data from the two platforms, to determine whether the data from each platform contains the same set of database modification events. If the data from each platform contains the same database modification events, the migration of the batch application is successful; otherwise, the migration is unsuccessful.

[00079] **Figure 6** further illustrates a system for determining the success of an application migration from a mainframe platform to an open platform, in accordance with embodiment.

[00080] In accordance with an embodiment, the test manager application can run on the rehosting platform or another open system, and can be associated with a rehosting database client 619 and a mainframe database client 613.

[00081] In accordance with an embodiment, the rehosting database can be an Oracle database and the mainframe database can be a DB2 database. In this implantation, the rehosting database client can an Oracle client configured to connect to the Oracle database, and the mainframe client can be a DB2 client configured to connect to the DB2 database.

[00082] In accordance with an embodiment, a trigger script for the mainframe database 615 and a trigger script for the rehosting database 617 can be generated, and executed via the database clients for the mainframe database and the rehosting database, to create a plurality of triggers 535 and a plurality of triggers 523 in each database.

[00083] **Figure 7** further illustrates a system for determining the success of an application migration from a mainframe platform to an open platform, in accordance with embodiment.

[00084] As shown in Figure 7, modification events data from each database can be downloaded 711, 713 to the client application, and can be provided to the modification events comparison logic component.

[00085] In accordance with an embodiment, the database modification events data from each database can be contained in a plain text file. Each plain text file can include a plurality of database events, for example, INSERT, UPDATE and DELETE, from each database.

[00086] Since the database modification events are from different database products, their format can be different. The modification events comparison logic component can include code to parse each plain text file, and format the modification events from each database, so that a meaningful comparison can be performed between the two plain text files to determine whether the same set of modification events have occurred on each platform.

[00087] In accordance with an embodiment, based on whether the execution of the one or more batch jobs displays the same behaviors on both platforms in terms of database modifications, the test manager application can determine whether the mainframe application has executed successfully on the open/rehosting platform, and thus the migration success of the mainframe application.

[00088] **Figure 8** illustrates a flowchart for determining the success of an application migration from a mainframe platform to an open platform, in accordance with embodiment.

[00089] As shown in Figure 8, at step 811, an application is migrated from a mainframe platform to a rehosting platform/open platform, and one or more tables associated with the application are replicated from a mainframe database of the mainframe platform, to a rehosting database of the rehosting platform.

[00090] In accordance with an embodiment, at this step, resources required by the application can be downloaded to a workbench from the mainframe platform. The resources can include JCL job scripts, SYSIN, PROC scripts, COBOL programs, COPYBOOK script, and DB2 DDL scripts. The downloaded resources can be merged and categorized, and can be placed into a workbench project.

[00091] Further, at this step, SQL scripts can be used to create corresponding tables on the rehosting (e.g., Oracle) database; and reload scripts can be used to reload the mainframe database (e.g., DB2 database) content into the rehosting database, so that the rehosting database can contain a full copy of the mainframe database data.

[00092] At step 813, a modification events table and associated sequence are created in

the mainframe database and the rehosting database using a client application.

[00093] At step 815, database trigger scripts are generated for the mainframe database and the rehosting database. A first set of trigger scripts are generated for creating database triggers for the one or more tables in the mainframe database, and a second set of triggers are generated for creating database triggers for the one or more tables in the rehosting database.

[00094] At step 817, the trigger scripts are executed to create database triggers in the mainframe database and the rehosting database.

[00095] At step 819, the batch job is submitted to the mainframe platform and the rehosting platform for execution, wherein the execution of the batch job causes the database triggers in each platform to store database modification events generated by the batch job into the modification events table in each platform.

[00096] At step 821, the return code of the batch job from the rehosting/open platform is checked to determine whether the return code indicates an error during the execution of the batch job in the rehosting platform. If the job execution returns an error code, the migration would be determined to be unsuccessful as shown at step 833.

[00097] However, if the job execution returns a warning or success code, the database modification events data is downloaded from each platform as shown at step 823.

[00098] At step 827, the database modification events data from the rehosting platform and the database events data from the mainframe platform are compared.

[00099] At step 829, the client application is used to determine whether the batch job execution generates the same set of database operations. If the same set of database operations are generated on each platform, the migration of the batch job is determined to be successful, as shown at step 831; otherwise, the job migration is determined to be a failure.

[000100] In accordance with an embodiment, when storing a data update to a BLOB data for a BLOB column in a modification events table, the BLOB data can be converted to readable hex character using hex() function provided by DB2. In accordance with an embodiment, the database triggers can capture the first 16352 bytes of the BLOB data update.

[000101] In accordance with an embodiment, the modification events tables on both platform can be cleaned for the next job execution, including dropping the triggers created previously, and cleaning the modification events data and associated sequence.

[000102] **Figure 9** illustrates a method for determining the success of an application migration from a mainframe platform to an open platform, in accordance with embodiment.

[000103] As shown by Figure 9, at step 911, a rehosting platform is provided on a computer including one or more microprocessors, wherein the rehosting platform includes an

application container for executing a batch job migrated from a mainframe platform.

[000104] At step 913, one or more database triggers and a modification events table are created in a first database associated with the rehosting platform, and a second database associated with the mainframe platform.

[000105] At step 915, the batch job is submitted from a client application to the rehosting platform and the mainframe platform for execution, where the batch job execution on each platform generates a plurality of database modification events.

[000106] At step 917, the database triggers on each platform are used to capture the database modification events, and store them to the modification events table on that platform.

[000107] At step 919, the database modification events from each database are downloaded to the client application, which is configured to determine a success of the batch job migration by comparing the database modification events from the first database and the database modification events from the second database.

[000108] The present invention may be conveniently implemented using one or more conventional general purpose or specialized digital computer, computing device, machine, or microprocessor, including one or more processors, memory and/or computer readable storage media programmed according to the teachings of the present disclosure. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art.

[000109] In some embodiments, the present invention includes a computer program product which is a non-transitory storage medium or computer readable medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the processes of the present invention. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

[000110] The foregoing description of the present invention has been provided for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to the practitioner skilled in the art. The modifications and variations include any relevant combination of the disclosed features. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various

modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalents.

Claims:

What is claimed is:

1. A system for determining the migration success of an application, comprising:
 - a computer that includes one or more microprocessors;
 - a rehosting platform, executing on the computer, that includes an application container for executing an application migrated from a mainframe platform;
 - a first database associated with the rehosting platform, and a second database associated with the mainframe platform, each database including one or more database triggers and a modification events table;
 - a client application configured to submit a batch job associated with the application to the rehosting platform and the mainframe platform for execution, where the batch job execution on each platform generates a plurality of database modification events;
 - wherein the database triggers on each platform capture the database modification events, and store them to the modification events table on that platform; and
 - wherein the client application is configured to download the database modification events from each database, and determine a success of the migration by comparing the database modification events from the first database and the database modification events from the second database.
2. The system of claim 1, wherein the client application is configured to determine whether the database modification events from the first database and the database modification events from the second database contain the same set of database operations.
3. The system of claim 1 or 2, wherein the database modification events from the first database and the database modification events from the second database are both downloaded in a plain text file, and wherein the client application includes a comparison logic configured to format each plain text files for comparison.
4. The system of any of claims 1 to 3, wherein the database triggers on each platform are generated by a database trigger script created in the client application.

5. The system of any of claims 1 to 4, wherein the first database is an Oracle database, and the second database is a DB2 database, and wherein the client application is configured to an Oracle client and a DB2 client to communicate with the first database and the second database.
6. The system of claim 1, further comprising
checking a return code of the batch job execution.
7. The system of any of claims 1 to 6, wherein the batch job is a Job Control Language (JCL) job, and the JCL job is submitted to the rehosting platform for execution without converting the JCL job to another language.
8. A method for determining the migration success of a mainframe batch job, comprising:
providing a rehosting platform, executing on the computer, that includes an application container for executing an application migrated from a mainframe platform;
creating one or more database triggers and a modification events table in a first database associated with the rehosting platform, and a second database associated with the mainframe platform;
submitting a batch job associated with the application, from a client application, to the rehosting platform and the mainframe platform for execution, where the batch job execution on each platform generates a plurality of database modification events;
using the database triggers on each platform to capture the database modification events, and store them to the modification events table on that platform; and
downloading the database modification events from each database to the client application, which is configured to determine a success of the application migration by comparing the database modification events from the first database and the database modification events from the second database.
9. The method of claim 8, wherein the client application is configured to determine whether the database modification events from the first database and the database modification events from the second database contain the same set of database operations.
10. The method of claim 8 or 9, wherein the database modification events from the first database and the database modification events from the second database are both downloaded

in a plain text file, and wherein the client application includes a comparison logic configured to format each plain text files.

11. The method of any of claims 8 to 10, wherein the database triggers on each platform are generated by a database trigger script created in the client application.

12. The method of any of claims 8 to 11, wherein the first database is an Oracle database, and the second database is a DB2 database, and wherein the client application is configured with an Oracle client and a DB2 client to communicate with the first database and the second database.

13. The method of claim 8, further comprising
checking a return code of the batch job execution.

14. The method of any of claims 8 to 13, wherein the batch job is a Job Control Language (JCL) job, and the JCL job is submitted to the rehosting platform for execution without converting the JCL job to another language.

15. A non-transitory computer-readable storage medium storing a set of instructions for determining the migration success of an application, said instructions, when executed by one or more processors, causing the one or more processors to perform the steps comprising:

providing a rehosting platform, executing on the computer, that includes an application container for executing an application migrated from a mainframe platform;

creating one or more database triggers and a modification events table in a first database associated with the rehosting platform, and a second database associated with the mainframe platform;

submitting a batch job associated with the application, from a client application, to the rehosting platform and the mainframe platform for execution, where the batch job execution on each platform generates a plurality of database modification events;

using the database triggers on each platform to capture the database modification events, and store them to the modification events table on that platform; and

downloading the database modification events from each database to the client application, which is configured to determine a success of the an application migration by

comparing the database modification events from the first database and the database modification events from the second database.

16. The non-transitory computer-readable storage medium of claim 15, wherein the client application is configured to determine whether the database modification events from the first database and the database modification events from the second database contain the same set of database operations.

17. The non-transitory computer-readable storage medium of claim 15 or 16, wherein the database modification events from the first database and the database modification events from the second database are both downloaded in a plain text file, and wherein the client application includes a comparison logic configured to format each plain text files for comparison.

18. The non-transitory computer-readable storage medium of any of claims 15 to 17, wherein the database triggers on each platform are generated by a database trigger script created in the client application.

19. The non-transitory computer-readable storage medium of any of claims 15 to 18, wherein the first database is an Oracle database, and the second database is a DB2 database , and wherein the client application is configured to an Oracle client and a DB2 client to communicate with the first database and the second database.

20. The non-transitory computer-readable storage medium of any of claims 15 to 19, wherein the batch job is a Job Control Language (JCL) job, and the JCL job is submitted to the rehosting platform for execution without converting the JCL job to another language.

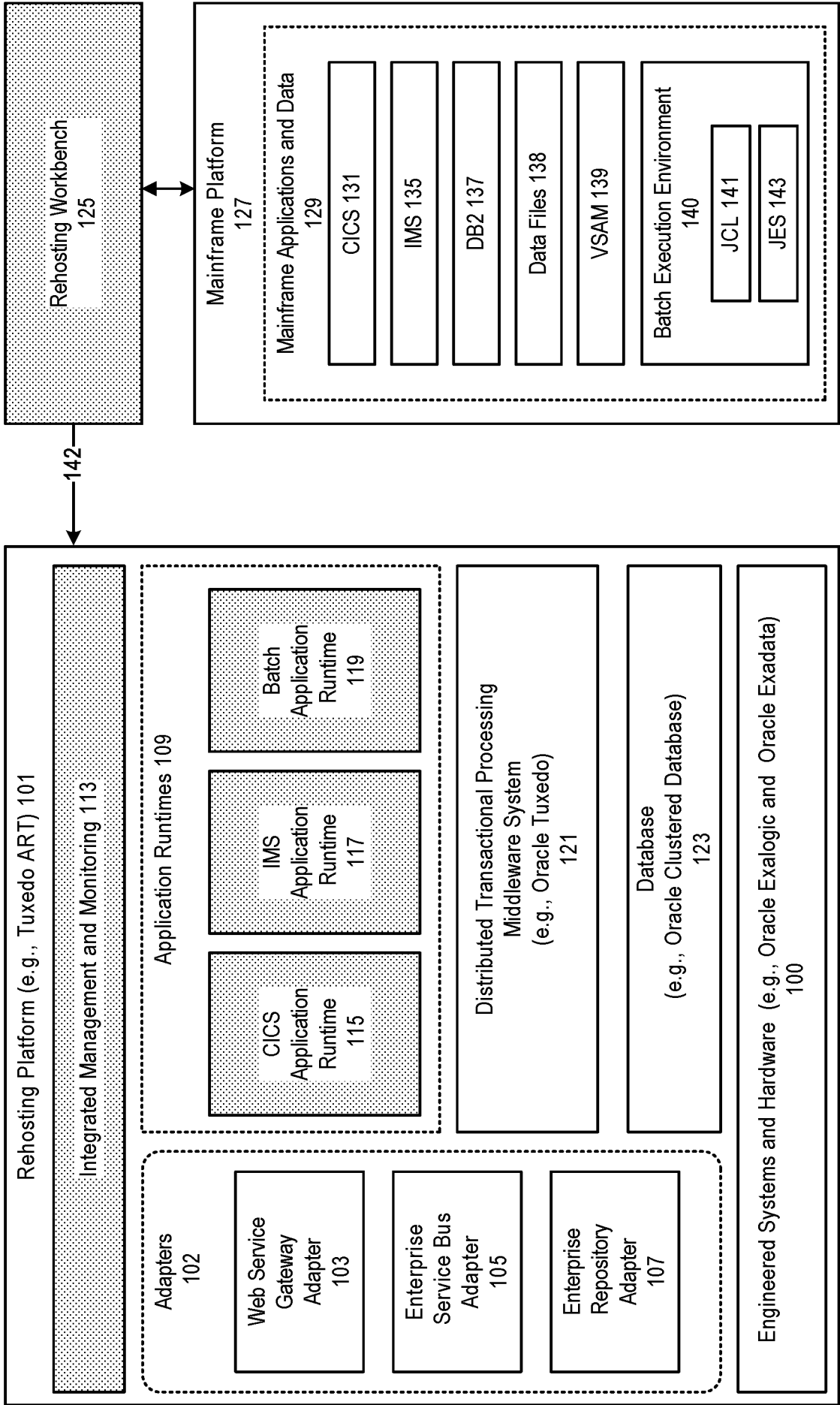


Figure 1

The BEA Tuxedo ATMI Basic Architecture

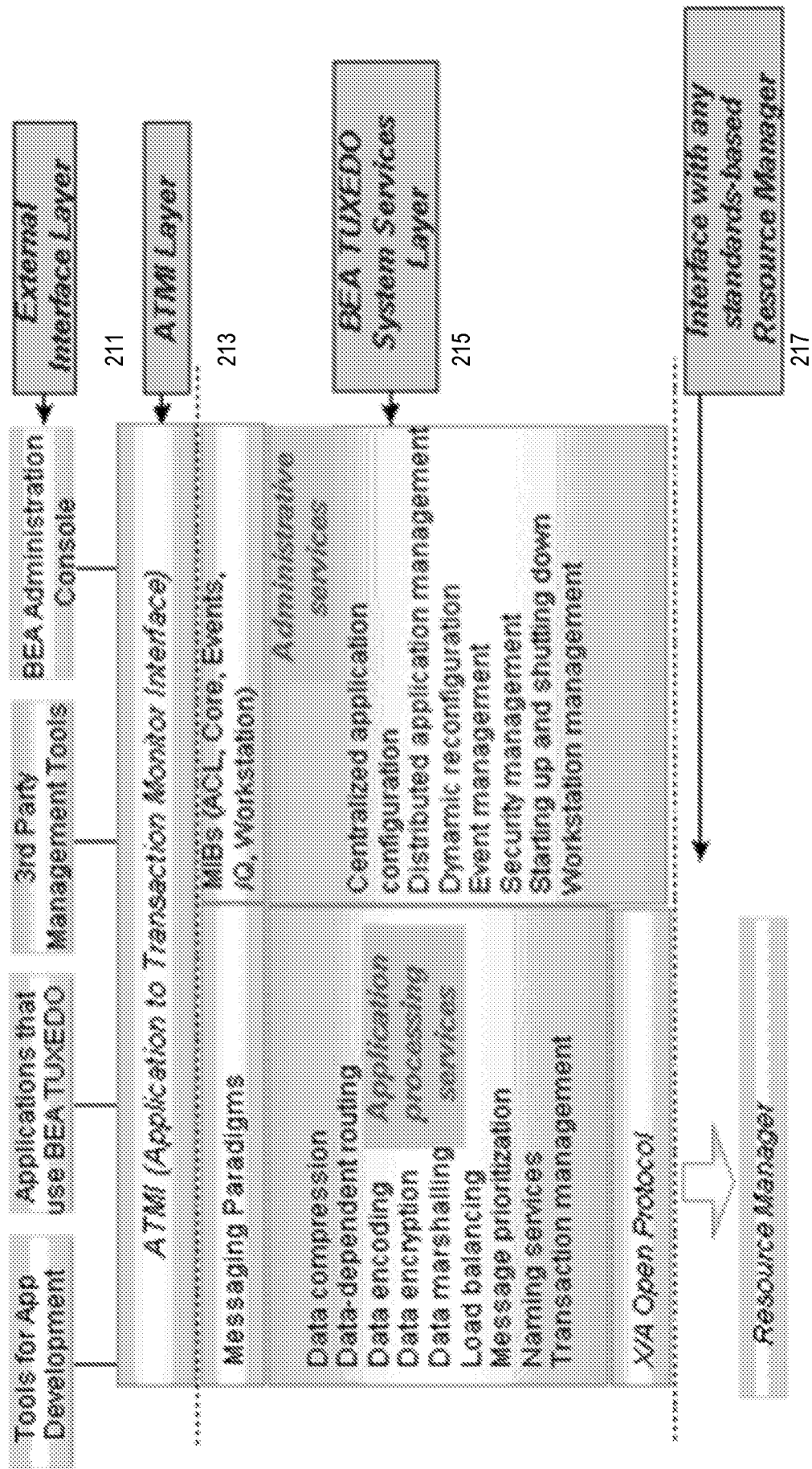


Figure 2

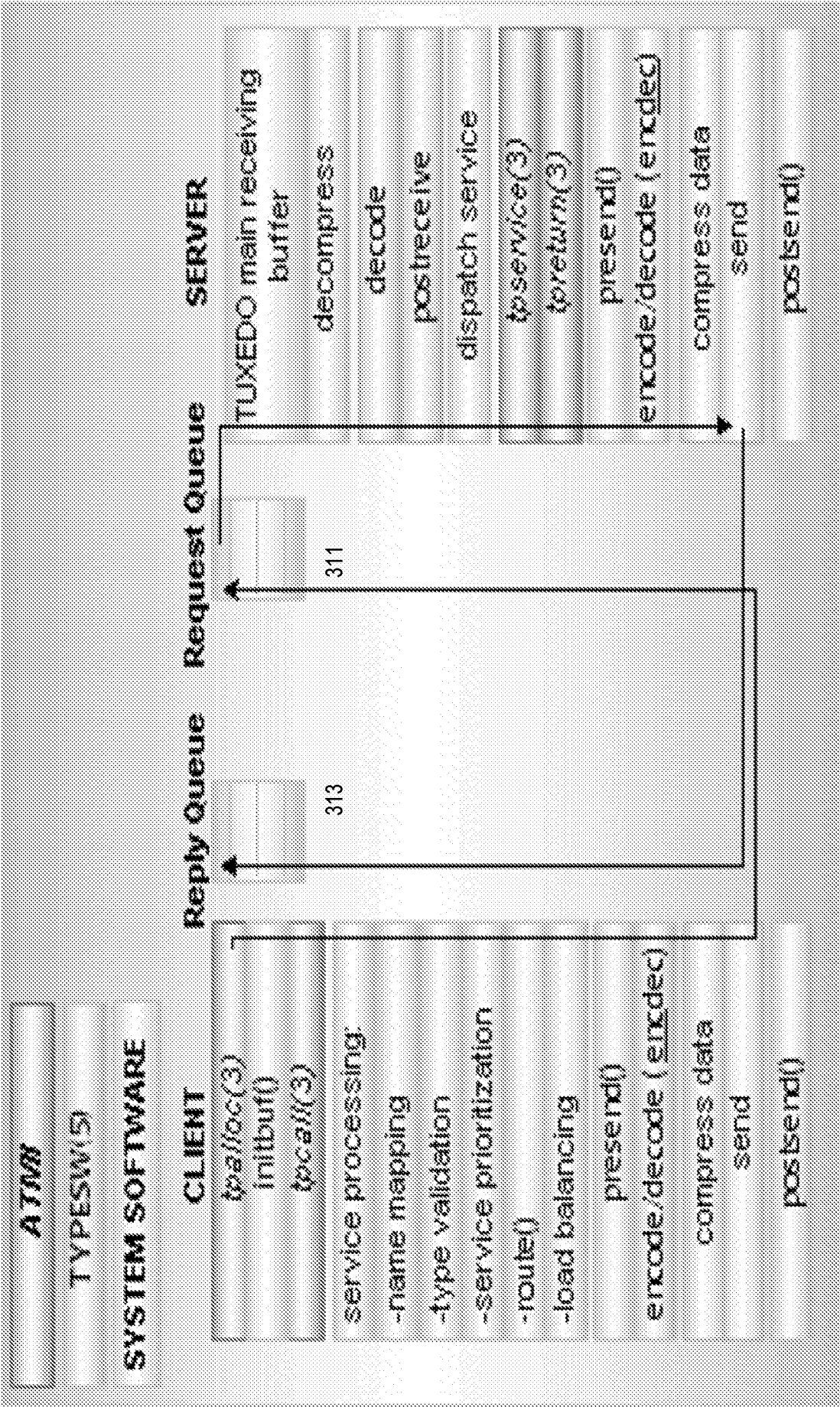


Figure 3

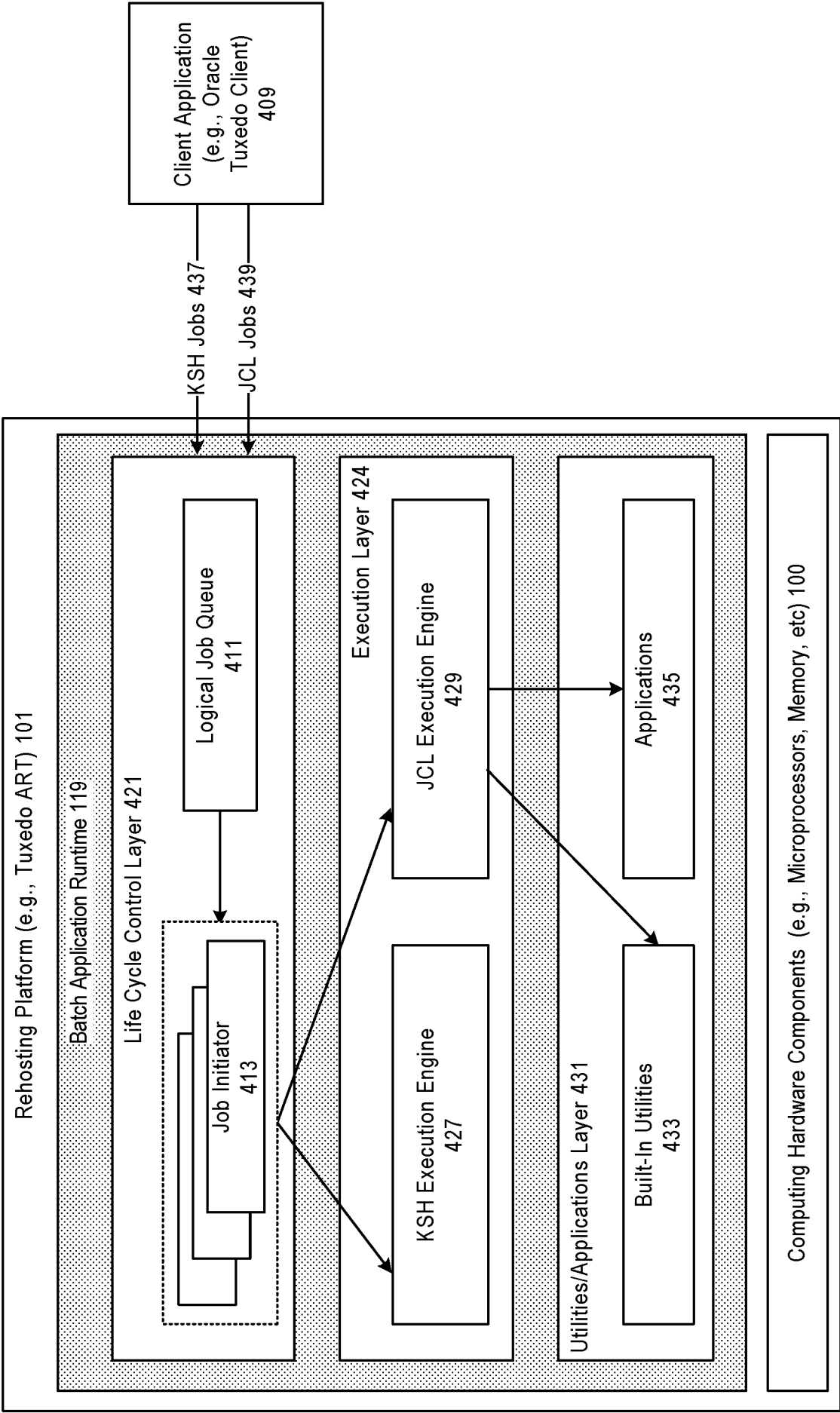


Figure 4

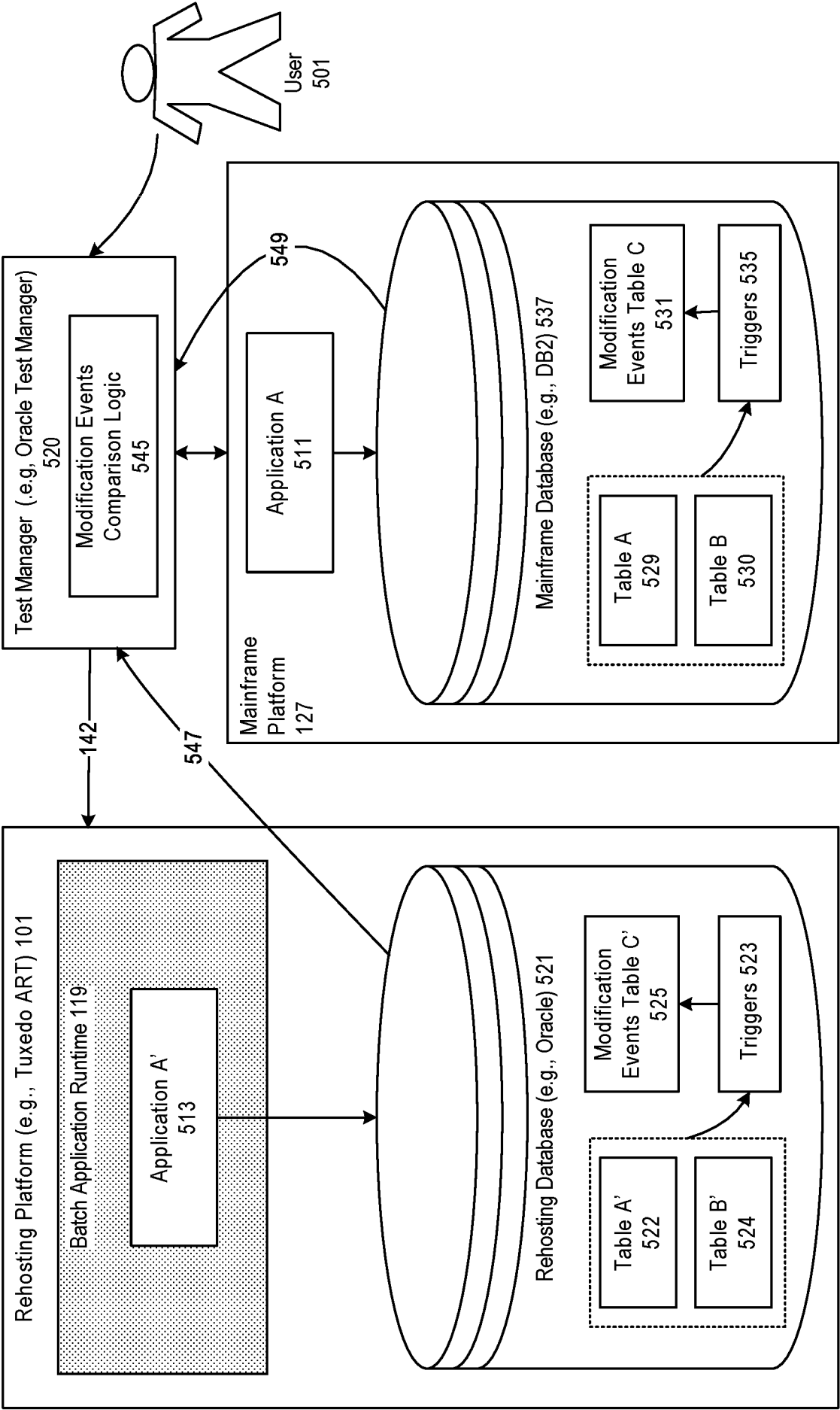


Figure 5

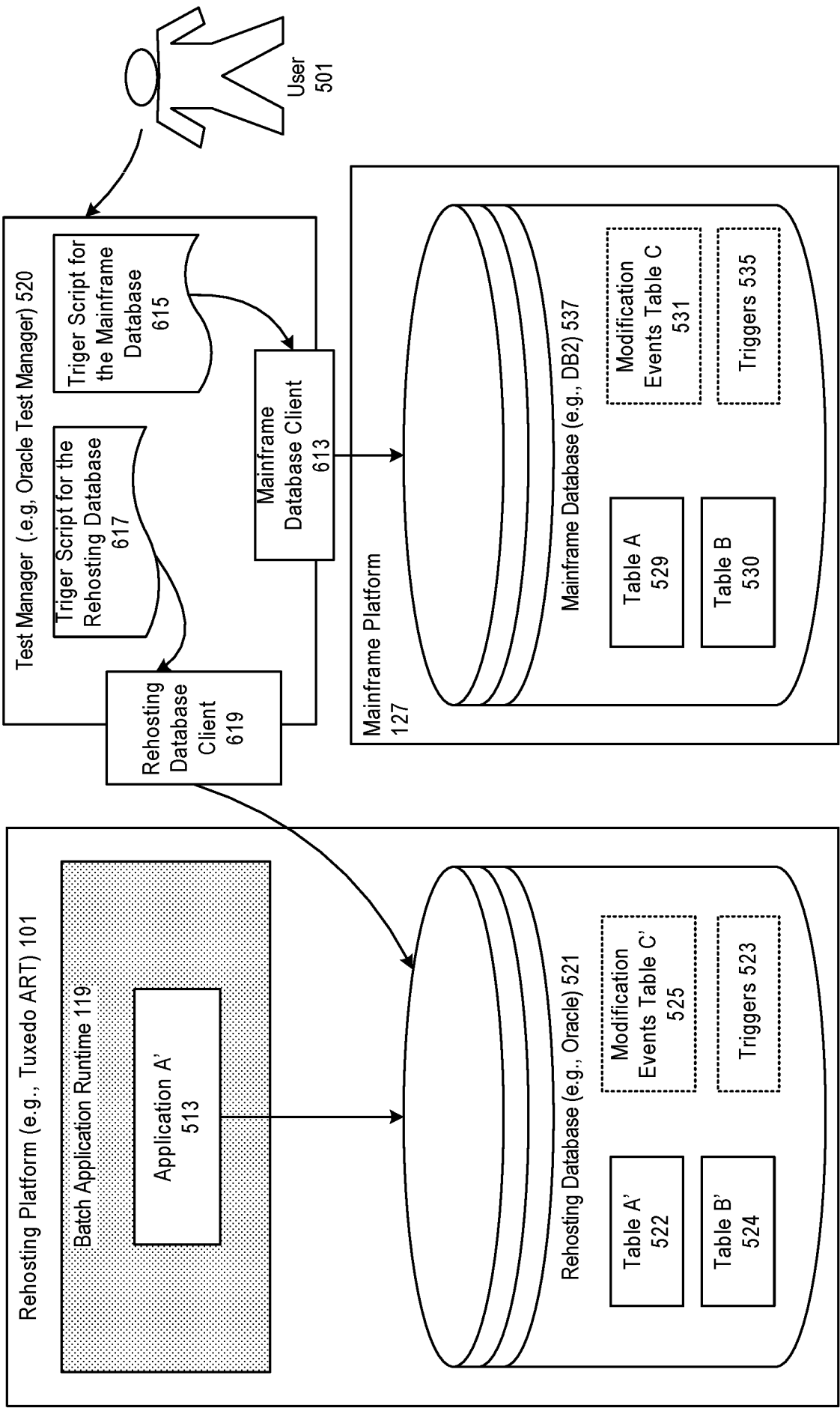


Figure 6

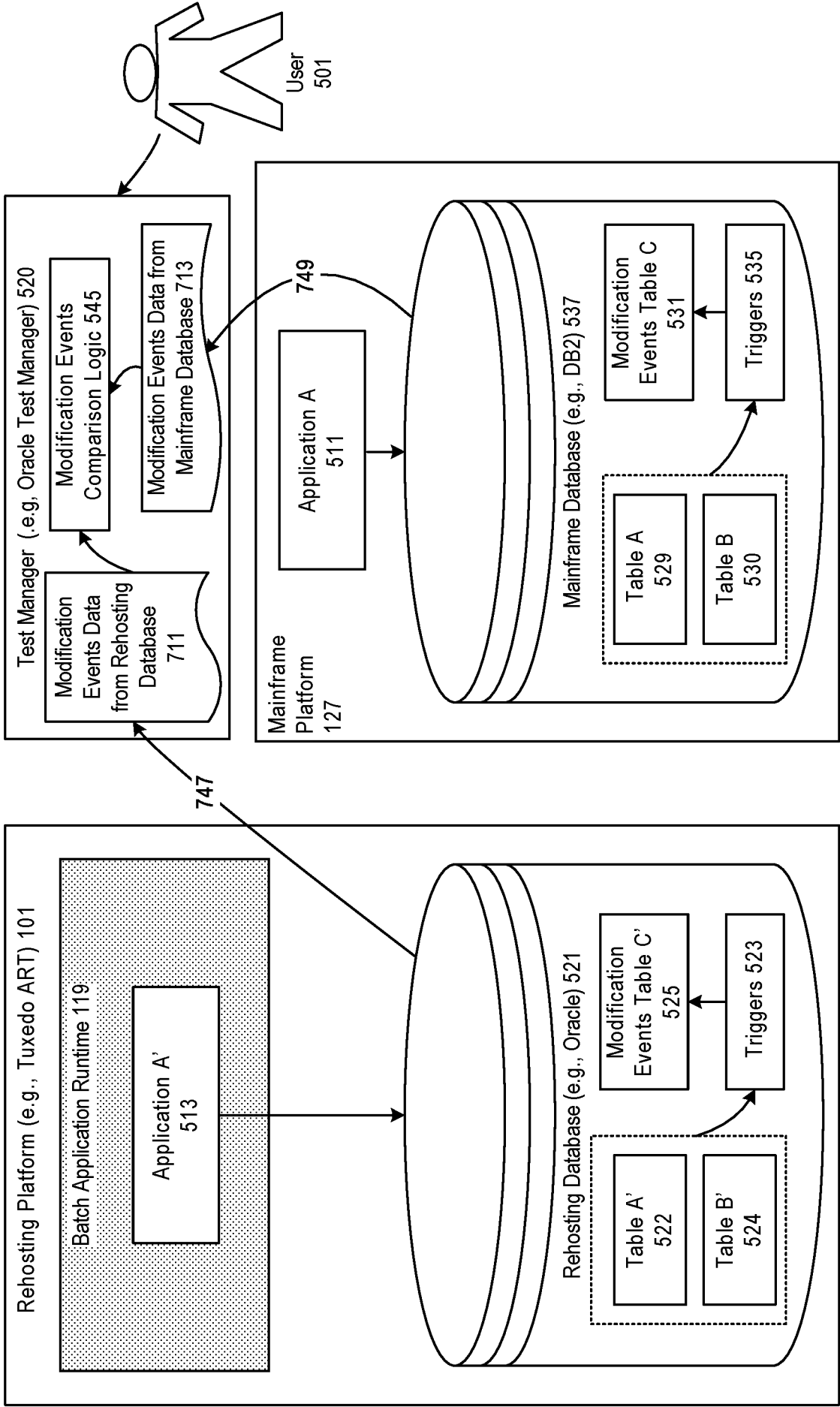


Figure 7

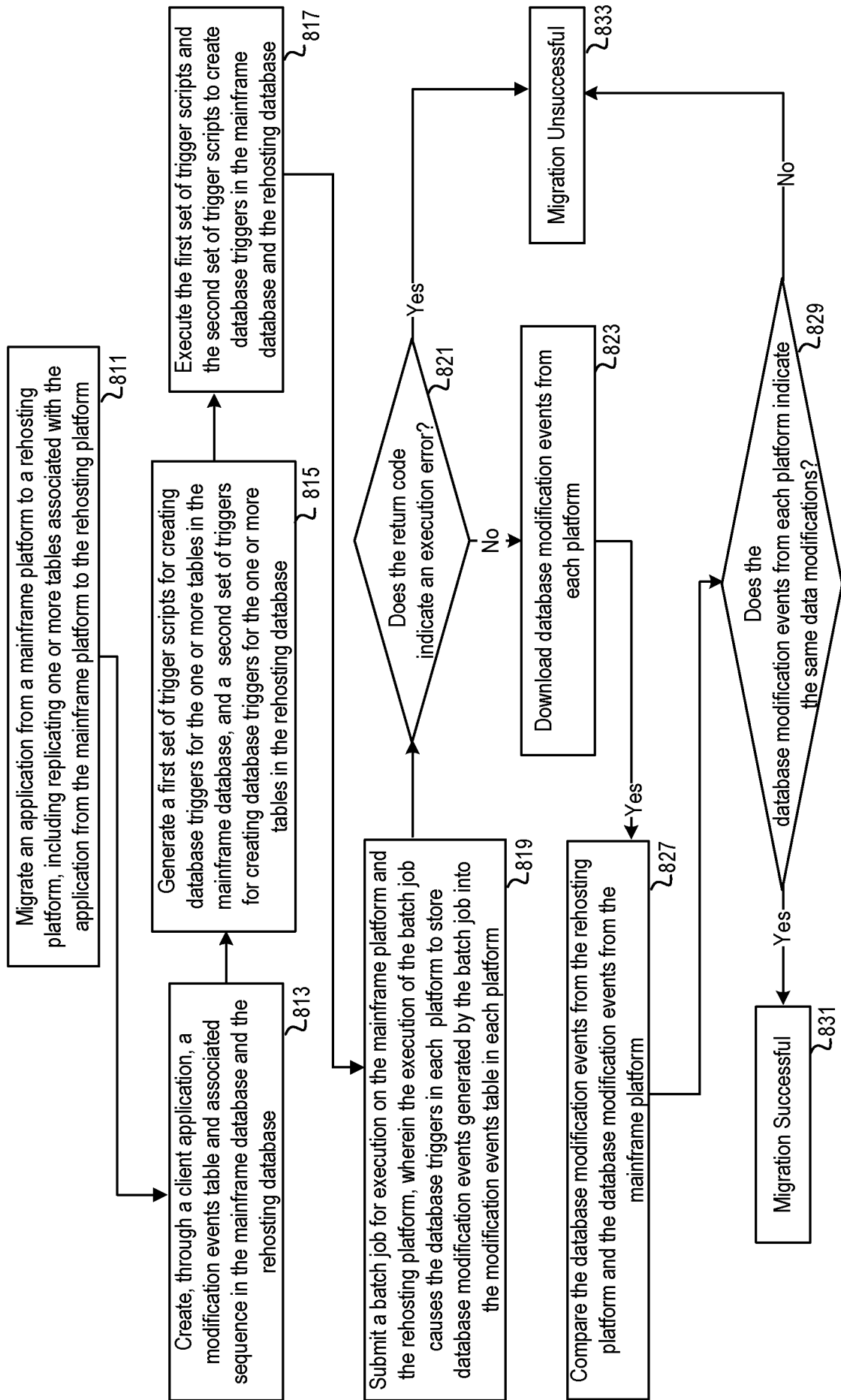


Figure 8

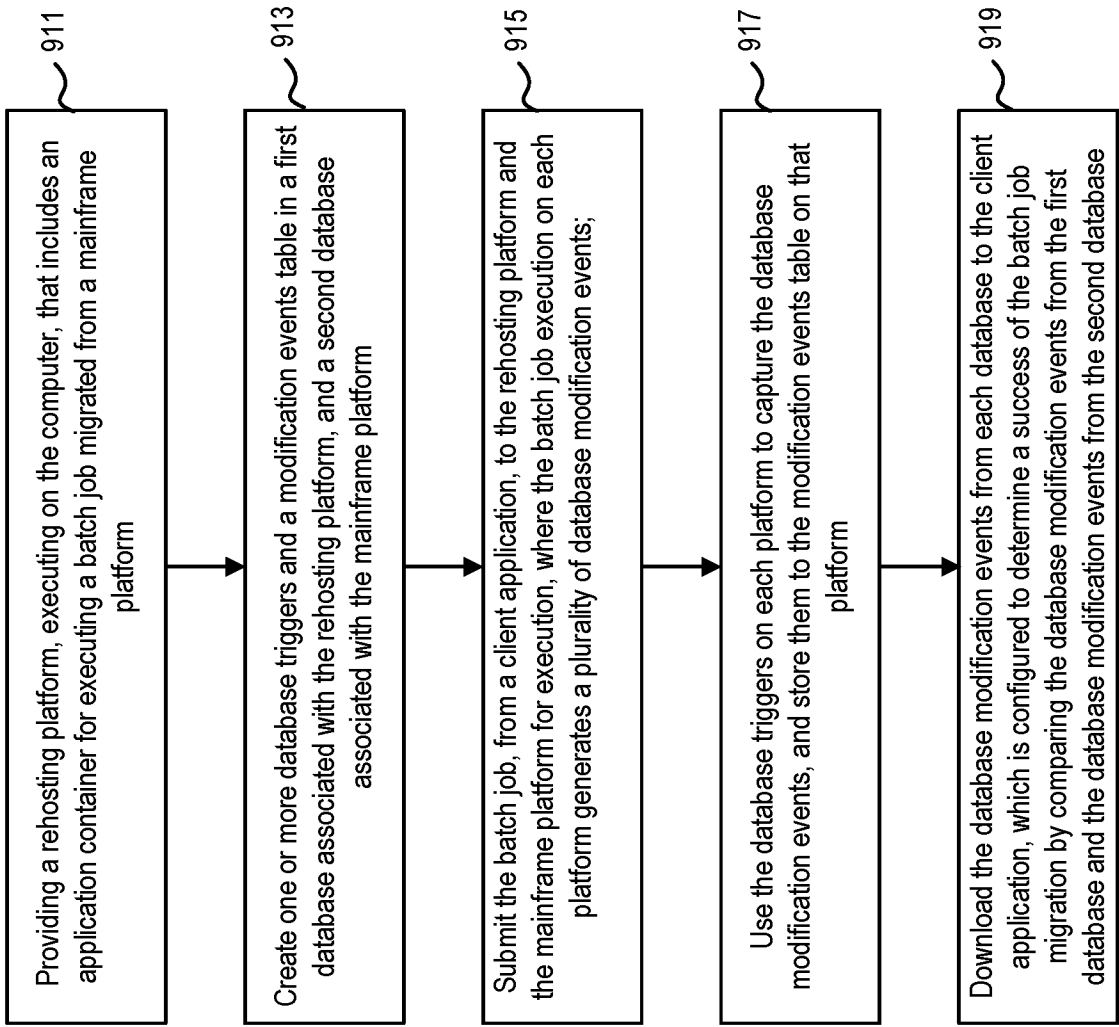


Figure 9

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2017/078914**A. CLASSIFICATION OF SUBJECT MATTER**

G06F 11/34(2006.01)i

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)

G06F 11/-

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 practicable, search terms used)

CNABS, DWPI, SIPOABS, CNTXT, CNKI: migration, application, success+, platform, database, compare+

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 101218565 A (UBS AG) 09 July 2008 (2008-07-09) description, page 2 line 26 to page 7 line 4, page 7 line 28 to page 17 line 3	1-20
A	CN 101727383 A (SHANGHAI MEDICAL INSURANCE INFORMATION C ET AL) 09 June 2010 (2010-06-09) the whole document	1-20
A	CN 104063320 A (SHANGHAI FEIXUN COMMUNICATION CO., LTD.) 24 September 2014 (2014-09-24) the whole document	1-20

☐ Further documents are listed in the continuation of Box C.☒ See patent family 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 application or patent 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

05 December 2017

Date of mailing of the international search report

27 December 2017

Name and mailing address of the ISA/CN

**STATE INTELLECTUAL PROPERTY OFFICE OF THE
P.R.CHINA**
**6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing
100088
China**

Facsimile No. (86-10)62019451

Authorized officer

ZHU, Xiaoli

Telephone No. (86-10)62411672

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2017/078914

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	101218565	A	09 July 2008	EP	1746502	A1	24 January 2007
				CN	101218565	B	11 January 2012
				WO	2007003432	A1	11 January 2007
				US	2007011209	A1	11 January 2007
				HK	1113418	A1	14 September 2012
				US	7831636	B2	09 November 2010
				SG	138837	B	31 December 2009
				SG	138837	A1	29 February 2008
				IN	200800478	P2	17 October 2008
CN	101727383	A	09 June 2010	CN	101727383	B	04 July 2012
CN	104063320	A	24 September 2014	None			