CLAIMS

WHAT IS CLAIMED IS:

1. A computer system for internal audit and internal control management, the system comprising:

a platform for integrated internal audit and internal control management, the platform comprising a development environment, a run-time environment, and a management environment;

a plurality of data sources stored in databases and related to auditable processes subject to the internal audit;

a plurality of repositories storing data associated with controls and rules related to the control management;

a plurality of interconnected computer subsystems having programming routines executable in at least one of said three environments, the computer subsystems comprising first, second, and third sets of subsystems;

wherein each of said subsystems for all of said three environments are user-accessible through a thin client comprising an internet browser, the thin client having a graphical user interface with selectable fields corresponding to each of said subsystems of each of said three environments simultaneously displayed on an associated user-perceptible screen;

wherein the first set of subsystems is executable in the development environment, the first set of subsystems comprising a data on-boarder, connection adapters, GRC forms, and a control designer;

wherein the second set of subsystems is executable in the run-time environment and comprises a data integration server;

wherein the third set of subsystems is executable in the management environment and comprises incident management, issue management, scheduling, monitoring, and security;

a RESTful application programming interface generated by, and having user-selectable fields associated with, the first set of subsystems in the development environment;

a plurality of dashboards generated by, and having user-selectable fields associated with, the third set of subsystems in the management environment;

wherein the dashboards comprise a control monitoring scoreboard, the scoreboard simultaneously displaying key performance indicators determined in real time by the third set of subsystems of the management environment, the key performance indicators comprising controls failed, related records scanned, tests performed, and associated risk scores, foregoing determinations displayed on the scoreboard as user-perceptible indicia;

the scoreboard having programming for displaying detail records of respective ones of the key performance indicators in response to user selection of an associated field on the scoreboard;

wherein the data on-boarder comprises programming capable of performing the following computer-implemented steps in response to user input through the RESTful application programming interface:

create a first data flow mapping corresponding to a first data source object to be on-boarded;

edit a second data flow mapping previously created by the data on-boarder, the second data flow mapping corresponding to a second data source object;

drag and drop at least one of the data source objects to be on-boarded into a first logical work space accessible from the RESTful application programming interface;

select a predetermined connection for the dropped source object from a plurality of the connections determined by the connection adapters;

generate a source qualifier and default select query for the dropped source object;

develop and validate the data flow mappings using transformation objects selected from the group consisting of Joiner, Filter, Lookup, Router, Cache, Expression, and JAVA/Scala/Python/R transformations;

connect to target definitions after validation of the data flow mappings;

execute the data flow mappings to generate associated run statistics, and

display in real-time first indicia corresponding to the execution of the data flow mappings;

wherein the steps of dragging and dropping and connecting to the target definitions are executable in a manner agnostic to script language associated with the source object and the target definitions;

wherein the control designer subsystem of the first set of subsystems comprises programming capable of performing the following computer-implemented steps, in response to user-input:

access one of the databases to retrieve an existing control name having a first control mapping associated therewith and defined in the GRC forms in a tree view folder associated with a control category;

edit a second control mapping previously designed by the control designer subsystem;

transfer at least one of the control mappings to a second logical workspace accessible from the RESTful application programming interface;

drag and drop at least one of the data source objects into the second logical workspace;

develop and validate at least one of the control mappings using transformation objects selected from the group consisting of control mapper, SQL control, and control result sets;

connect to a control result set after validation of at least one of the control mappings;

execute the control mappings to generate a log associated with the controls of the executed control mappings;

display in real-time second indicia corresponding to the execution of the control mappings;

wherein the control mapper comprises programming capable of maintaining control logic associated with the corresponding one of the control mappings, irrespective of substitution of one of the data source objects of the corresponding control mapping with another one of the data source objects retrieved from the databases.

2. The system of claim 1, wherein the data on-boarder subsystem and the control designer subsystem consist essentially of programming in a micro service architecture.

3. The system of claim 1, wherein the control monitoring scoreboard comprises programming for displaying on a single, user-perceptible display screen indicia corresponding to performance of the controls being monitored by the system, the performance quantified by

numbers determined for predetermined, corresponding periods of time, the displayed indicia corresponding to data determined by the control monitor and comprising functions, processes, risks, controls, completed and failed monitoring, incidents by status, batches executed by status, tasks executed by status, and issues by status.

4. The system of claim 1, wherein the data on-boarder comprises programming, which, when executed, configures the data source objects into logical columns in the data flow mapping, the programming displaying indicia corresponding to the logical columns in the RESTful API, and further including programming to access data records associated with the data source objects corresponding to the logical columns through pipelined linkages.

5. The system of claim 1, wherein the connection adaptors comprise:

a connections interface;

programming executable in response to user selection through the connections interface and capable of performing, when executed, the steps of:

access the data sources, wherein the data sources are heterogeneous and comprise relational data sources, non-relational data sources, data related to ERP applications, data related to non-ERP applications, discrete ones of the data sources formatted in conformance with respective industry standards;

wherein the step of drag and drop of the data source object is performed by programming to receive any of the data source objects from the heterogeneous data sources, irrespective of the respective industry standards to which the data source object conforms and to place the data source objects into the first logical work space by means of one of the connection adaptors corresponding to a respective one of the heterogeneous data sources.

6. The system of claim 1, wherein the scheduling subsystem comprises programming, when executed by the user, capable of performing the following steps:

through a single user-perceptible display screen, enabling selection and scheduling of applications, batches, and tasks, the programming permitting scheduling in real time, the programming being capable of scheduling applications, batches, and tasks in any of the time intervals comprising continuous, intermittent, and one-time;

the programming determining the results of the scheduling and displaying and updating indicia corresponding to the results of the scheduling.

7. The system of claim 1, wherein the data on-boarder and the control designer have programming capable of generating data flow mapping metadata and control mapping metadata, respectively, the programming storing the metadata in one of the databases accessible to the data integration server, the programming generating each of the data flow mappings and the control mappings in response to receiving user-input in a single, domain-specific programming language selected from the group consisting of JAVA, Python, and Scala.

8. The system of claim 1, wherein the monitoring system comprises monitoring programming to continuously monitor the controls in real-time through a single, user-perceptible monitor user interface, the monitoring programming, when executed, capable of performing the following steps:

identify in real-time a failure during execution of a control-related task;

generate an incident report accessible by the incident management system;

upon resolution of the incident, cause a restart of the control-related task at an audit control point associated with the failure;

wherein the monitoring program is capable of:

identifying scheduled, continuous, and real-time batches, tasks, and a history of task executions;

displaying corresponding indicia on the monitor user interface in at least one of a monitoring view and a task view;

updating the indicia in real-time, the indicia having user-selectable fields associated therewith; and

retrieving task details in response to user selection, whereby the controls are continuously monitored by the monitoring programming.

9. The system of claim 1, wherein the GRC forms subsystem comprises programming, when executed in response to user selection, capable of performing the steps of:

launch a GRC forms user interface;

access data corresponding to controls and rules from the controls and rules repositories, respectively, to structure discrete ones of the rules into at least one corresponding control to create entries into a GRC form;

validate operation of the entries with selected ones of the data objects;

if the validation is successful, generate a corresponding new or updated one of the connection adapters; and

link the GRC form to the rules repository.

10. A computer-implemented method of internal control management, including steps executed by a computer processor implementing computer executable instructions, the steps, comprising:

providing a plurality of dashboards generated by, and having user-selectable fields associated with the internal control management, one of the dashboards comprising a control monitoring scoreboard;

simultaneously displaying on the scoreboard key performance indicators determined in real time, the key performance indicators comprising controls failed, related records scanned, tests performed, and associated risk scores;

displaying foregoing determinations on the scoreboard as user-perceptible indicia;

displaying detail records of respective ones of the key performance indicators in response to user selection of an associated field on the scoreboard;

on-boarding data by performing the following computer-implemented steps in response to user input through a RESTful application programming interface:

at least one of creating a first data flow mapping corresponding to a first data source programming interface source object to be on-boarded and editing a previously created, second data flow mapping corresponding to a second data source object;

dragging and dropping at least one of the data source objects to be on-boarded into a first logical work space accessible from the RESTful application programming interface;

selecting a predetermined connection for the dropped data source object from a plurality of the connections; and

generating a source qualifier and default select query for the dropped data source object;

executing the data flow mappings to generate associated run statistics, and

displaying in real-time first indicia corresponding to the execution of the data flow mappings;

wherein the steps of creating the first data flow mapping and editing the second data flow mapping comprise:

developing and validating the data flow mappings using transformation objects selected from the group consisting of Joiner, Filter,

Lookup, Router, Cache, Expression, and JAVA/Scala/Python/R transformations and connecting to target definitions after validation of the data flow mappings;

wherein the steps of dragging and dropping and connecting to the target definitions are executable in a manner agnostic to script language associated with the source object and the target definitions;

accessing one of a plurality of databases to retrieve an existing control name having a first control mapping associated therewith and defined in a GRC form in a tree view folder associated with a control category;

transferring the control mapping to a second logical workspace accessible from the RESTful application programming interface;

dragging and dropping at least one of the data source objects into the second logical workspace;

developing and validating the control mapping using transformation objects selected from the group consisting of control mapper, SQL control, and control result sets;

connecting to a control result set after validation of the control mapping;

executing the control mapping to generate a log associated with controls associated with the executed control mapping;

displaying in real-time second indicia corresponding to the execution of the control mapping;

maintaining control logic associated with the control mapping, irrespective of substitution of one of the data source objects of the control mapping with another one of the data source objects retrieved from the databases.

11. The method of claim 10, further comprising the steps of displaying indicia on a single, user-perceptible display screen, the indicia corresponding to performance of the controls being monitored by the system, the performance quantified by numbers determined for predetermined, corresponding periods of time, the displayed indicia corresponding to data determined by the control monitor, the quantitative data comprising functions, processes, risks, controls, completed and failed monitoring, incidents by status, batches executed by status, tasks executed by status, and issues by status.

12. The method of claim 10, further comprising the steps of configuring the data source objects into logical columns in the data flow mapping, displaying indicia corresponding to the logical columns in the RESTful API, and accessing data records associated with the data source objects corresponding to the logical columns through pipelined linkages.

13. The method of claim 10, further comprising the steps of:

providing a connections interface to access data sources, wherein the data sources are heterogeneous and comprise relational data sources, non-relational data sources, data related to ERP applications, data related to non-ERP applications, discrete ones of the data sources formatted in conformance with respective industry standards;

wherein the step of drag and drop of the data source object is performed by programming to receive any of the data source objects from the heterogeneous data sources, irrespective of the respective industry standards to which the data source object conforms and to place the data source objects into the first logical work space by means of respective ones of the connection adaptors corresponding to a respective one of the heterogeneous data sources.

14. The method of claim 10, further comprising the steps of:

enabling selection and scheduling of applications, batches, and tasks in real time and in any of the time intervals comprising continuous, intermittent, and one-time;

determining the results of the scheduling and displaying and updating indicia corresponding to the results of the scheduling through a single, user-perceptible display screen.

15. The method of claim 10, wherein the step of on-boarding comprises generating data flow mapping metadata and wherein the step of developing the control mapping comprises generating control mapping metadata, the generations of metadata for a plurality of the data flow mappings and control mappings performed by instructions in a plurality of domain-specific programming language selected from the group consisting of JAVA, Python, and Scala.

16. The method of claim 10, comprising the steps of:

continuously monitoring the controls in real-time through a single, user-perceptible monitor user interface;

identifying in real-time a failure during execution of a control-related task;

generating an incident report in user-perceptible form; and

upon resolution of the incident, causing a restart of the control-related task at an audit control point associated with the failure.

17. The method of claim 10, further comprising the steps of:

identifying scheduled, continuous, and real-time batches, tasks, and a history of task executions;

displaying in user-perceptible form indicia corresponding to the batches and tasks;

updating the indicia in real-time to continuously monitor the controls, the indicia having user-selectable fields associated therewith.

18. The method of claim 10, further comprising the steps of:

launching a GRC forms user interface;

accessing data corresponding to controls and rules to structure discrete ones of the rules into at least one corresponding control to create entries into a GRC form;

validating operation of the entries with selected ones of the data objects;

if the validation is successful, generating a corresponding connection adapters; and

linking the GRC form to a rules repository.