

1 APPARATUSES, METHODS AND SYSTEMS FOR PROVIDING A
2 VIRTUAL DEVELOPMENT AND DEPLOYMENT ENVIRONMENT
3 INCLUDING REAL AND SYNTHETIC DATA

4 RELATED APPLICATIONS

5 [0001] The instant application describes inventive aspects of at least two distinct
6 implementations, including:

7 [0002] a method for providing a virtual development and/or test environment
8 (with a suggested Class/Subclass of 726/26);

9 [0003] a method for data synthesization (with a suggested Class/Subclass of
10 703/22);

11 [0004] The instant application directs claims to Prevention Of Unauthorized Use
12 Of Data Including Prevention Of Piracy, Privacy Violations, Or Unauthorized Data
13 Modification (suggested class/subclass: 726/26). However, in order to develop a
14 reader's understanding of the the descriptions of the other implementations, the
15 implementations have been compiled into a single application to illustrate and clarify
16 how aspects of these implementations operate independently, interoperate as between
17 individual embodiments, and/or cooperate collectively. The application goes on to
18 further describe the interrelations and synergies as between any of the various
19 implementations within the context of an overarching inventive system; all of which is
20 to further ensure compliance with 35 U.S.C. §112.

[0005] Furthermore, applicants hereby claims priority under 35 USC §119 for United States provisional patent application serial no. 61/095.009 filed September 8, 2008 entitled "SYSTEM AND METHOD FOR PROVIDING A VIRTUAL BANKING ENVIRONMENT INCLUDING SYNTHETIC DATA."

[0006] The entire contents of the aforementioned application are herein expressly incorporated by reference.

FIELD

[0007] The present invention is directed generally to apparatuses, methods, and systems of computer testing, and more particularly, to APPARATUSES, METHODS AND SYSTEMS FOR PROVIDING A VIRTUAL DEVELOPMENT AND DEPLOYMENT ENVIRONMENT INCLUDING REAL AND SYNTHETIC DATA.

BACKGROUND

[0008] Software testing is utilized by software developers to reduce programming errors in software applications. Integrated development environments (IDEs), such as Microsoft Visual Studio and Eclipse, and the debuggers included with such IDEs are one set of tools that may be used by software developers for software testing.

SUMMARY

The APPARATUSES, METHODS AND SYSTEMS FOR PROVIDING A VIRTUAL DEVELOPMENT AND DEPLOYMENT ENVIRONMENT INCLUDING REAL AND SYNTHETIC DATA (hereinafter "SYNONYM") provides a virtual development and deployment environment. In one embodiment, a method involves receiving request to create synthesized testing data. Real data records to be anonymized are identified and a source of conjured data for anonymization of real data records based on a unique identifier in the real data records is selected. The method involves creating an anonymized data set by replacing real data in the real data records, based on the unique identifier in the real data records, with the selected conjured data, and selecting a subset of the anonymized data set based on specified criteria for synthesization. The method further involves generating a synthetic database, wherein generation further includes: instantiating a shell synthetic database, populating the shell synthetic data by copying the selected subset of the anonymized data set into the shell synthetic database, generating synthetic unique identifiers, and replacing unique identifiers with synthetic unique identifiers, and instantiating the synthetic database.

[0008a] According to a first aspect, there is provided a data synthesizing processor-implemented method, comprising: receiving a request to create synthesized testing data; identifying real data records for anonymization; selecting a source of conjured data for anonymization of real data records based on a unique identifier in the real data records; creating an anonymized data set by replacing real data in the real data records, based on the unique identifier in the real data records, with the selected conjured data; selecting a subset of the anonymized data set based on specified criteria for synthesization; generating a synthetic database, wherein generation further includes: instantiating a shell synthetic database, populating the shell synthetic data by copying the selected subset of the anonymized data set into the shell synthetic database, generating synthetic unique identifiers, and replacing unique identifiers with synthetic unique identifiers; and instantiating the synthetic database.

[0008b] According to a second aspect, there is provided a data synthesizing apparatus, comprising: a memory; a processor disposed in communication with said memory, and configured to issue a plurality of processing instructions stored in the memory, wherein the processor issues instructions to: receive a request to create synthesized testing data; identify real data records for anonymization; select a source of conjured data for anonymization of real data records based on a unique identifier in the real data records; create an anonymized data set by replacing real data in the real data records, based on the unique identifier in the real data records, with the selected conjured data; select a subset of the anonymized data set based on specified criteria for synthesization; generate a synthetic database, wherein generation further includes instructions to: instantiate a shell synthetic database, populate the shell synthetic data by copying the selected subset of the anonymized data set into the shell synthetic database, generate synthetic unique identifiers, and replace unique identifiers with synthetic unique identifiers; and instantiate the synthetic database.

[0008c] According to a third aspect, there is provided a data synthesizing processor-readable medium storing processor-issuable instructions to: receive a request to create synthesized testing data; identify real data records for anonymization; select a source of conjured data for anonymization of real data records based on a unique identifier in the real data records; create an anonymized data set by replacing real data in the real data records, based on the unique identifier in the real data records, with the selected conjured data; select a subset of the anonymized data set based on specified criteria for synthesization; generate a synthetic database, wherein generation further includes instructions to: instantiate a shell synthetic database, populate the shell synthetic data by copying the selected subset of the anonymized data set into the shell synthetic database, generate synthetic unique identifiers, and replace unique identifiers with synthetic unique identifiers; and instantiate the synthetic database.

[0008d] According to a fourth aspect, there is provided a data synthesizing system, comprising means to: means to receive a request to create synthesized testing data; means to identify real data records for anonymization; means to select a source of conjured data for anonymization of real data records based on a unique identifier in the real data records; means to create an anonymized data set by replacing real data in the real data records, based on the unique identifier in the real data records, with the selected conjured data; means to select a subset of the anonymized data set based on specified criteria for synthesization; means to generate a synthetic database, wherein generation further includes: means to instantiate a shell synthetic database, means to populate the shell synthetic data by copying the selected subset of the anonymized data set into the shell synthetic database, means to generate synthetic unique identifiers, and means to replace unique identifiers with synthetic unique identifiers; and means to instantiate the synthetic database.

[0008e] According to a fifth aspect, there is provided an environment testing processor-implemented method, comprising: receiving a request to create a test environment; identifying real data records based on the request; generating anonymized data from the real data; selecting a subset of the anonymized data based on the request; generating synthesized data from the subset of anonymized data; generating a synthetic test environment, wherein generation further includes: instantiating user accounts, populating the test environment with a profile of testing components, loading the synthesized data into the test environment; and deploying the synthetic test environment to a location specified in the request.

[0008f] According to a sixth aspect, there is provided an environment testing apparatus, comprising: a memory; a processor disposed in communication with said memory, and configured to issue a plurality of processing instructions stored in the memory, wherein the processor issues instructions to: receive a request to create a test environment; identify real data records based on the request; generate anonymized data from the real data; select a subset of the anonymized data based on the request; generate synthesized data from the subset of anonymized data; generate a synthetic test environment, wherein generation further includes instructions to: instantiate user accounts, populate the test environment with a profile of testing components, load the synthesized data into the test environment; and deploy the synthetic test environment to a location specified in the request.

[0008g] According to a seventh aspect, there is provided an environment testing processor-readable medium storing processor-issuable instructions to: receive a request to create a test environment; identify real data records based on the request; generate anonymized data from the real data; select a subset of the anonymized data based on the request; generate synthesized data from the subset of anonymized data; generate a synthetic test environment, wherein generation further includes instructions to: instantiate user accounts, populate the test environment with a profile of testing components, load the synthesized data into the test environment; and deploy the synthetic test environment to a location specified in the request.

[0008h] According to an eighth aspect, there is provided an environment testing system, comprising means to: means to receive a request to create a test environment; means to identify real data records based on the request; means to generate anonymized data from the real data; means to select a subset of the anonymized data based on the request; means to generate synthesized data from the subset of anonymized data; means to generate a synthetic test environment, wherein generation further includes: means to instantiate user accounts, means to populate the test environment with a profile of testing components, means to load the synthesized data into the test environment; and means to deploy the synthetic test environment to a location specified in the request.

[0008i] According to a ninth aspect, there is provided an environment deployment processor-implemented method, comprising: receiving a request to create a deployment environment; identifying real data records based on the request; generating varianced data from the real data; generating a synthetic deployment environment, wherein generation further includes: instantiating user accounts, populating the synthetic deployment environment with a profile of deployment components, loading the varianced data into the synthetic deployment environment; and deploying the synthetic deployment environment to a location specified in the request.

[0008j] According to a tenth aspect, there is provided an environment deployment apparatus, comprising: a memory; a processor disposed in communication with said memory, and configured to issue a plurality of processing instructions stored in the memory, wherein the processor issues instructions to: receive a request to create a deployment environment; identify real data records based on the request; generate varianced data from the real data; generate a synthetic deployment environment, wherein generation further includes instructions to: instantiate user accounts, populate the synthetic deployment environment with a profile of deployment components, load the varianced data into the synthetic deployment environment; and deploy the synthetic deployment environment to a location specified in the request.

[0008k] According to an eleventh aspect, there is provided an environment deployment processor-readable medium storing processor-issuable instructions to: receive a request to create a deployment environment; identify real data records based on the request; generate varianced data from the real data; generate a synthetic deployment environment, wherein generation further includes instructions to: instantiate user accounts, populate the synthetic deployment environment with a profile of deployment components, load the varianced data into the synthetic deployment environment; and deploy the synthetic deployment environment to a location specified in the request.

[0008l] According to a twelfth aspect, there is provided an environment deployment system, comprising means to: means to receive a request to create a deployment environment; means to identify real data records based on the request; means to generate varianced data from the real data; means to generate a synthetic deployment environment, wherein generation further includes: means to instantiate user accounts, means to populate the synthetic deployment environment with a profile of deployment components, means to load the varianced data into the synthetic deployment environment; and means to deploy the synthetic deployment environment to a location specified in the request.

[0008m] According to a thirteenth aspect, there is provided a processor-implemented method for synthesizing data, comprising: receiving a request to create synthesized customer records; identifying real data records for anonymization; selecting a source of conjured data for the anonymization of real data records; replacing real data in the real data records with the selected conjured data; retrieving anonymized customer records to be synthesized; creating synthetic customer records with synthetic customer record identifiers to hold anonymized data contained in the anonymized customer records; copying the anonymized data into the synthetic customer records; locating, via a processor, in the copied anonymized data customer record identifiers; replacing the located customer record identifiers with the synthetic customer record identifiers; and outputting the synthesized customer records.

[0008n] According to a fourteenth aspect, there is provided a system for facilitating development and testing of software applications using synthesized customer records comprising: a server; a software application for development and testing; a development environment running on the server, wherein the development environment is generated by cloning portions of a production environment that are required to run the software application; a database containing synthesized customer records, wherein: the database is communicatively connected with the development environment; and the database provides a source of data for the development environment.

[0008o] According to a fifteenth aspect, there is provided a method for development and testing of software applications using synthesized customer records comprising: connecting to a development environment with a workstation; providing authentication information required to access the development environment; and testing, on the workstation, a software application, wherein testing the software application involves accessing synthesized customer records.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The accompanying appendices and/or drawings illustrate various non-limiting, example, inventive aspects in accordance with the present disclosure:

[0010] FIGURE 1A is of a block diagram illustrating the creation and use of a virtual development and/or test environment in one embodiment of the SYNONYM;

1 **[0011]** FIGURE 1B is of a data flow diagram illustrating the transformation of real
2 data into synthetic data in one embodiment of the SYNONYM;

3 **[0012]** FIGURE 1C is of screen shot diagram illustrating a web based user
4 interface in one embodiment of the SYNONYM;

5 **[0013]** FIGURE 2A is of a logic flow diagram illustrating creation of anonymized
6 data in one embodiment of the SYNONYM;

7 **[0014]** FIGURE 2B is a table illustrating anonymization rules in one embodiment
8 of the SYNONYM;

9 **[0015]** FIGURE 2C is of screen shot diagram illustrating a web form based user
10 interface in one embodiment of the SYNONYM;

11 **[0016]** FIGURE 3 is of a logic flow diagram illustrating creation of synthesized
12 data in one embodiment of the SYNONYM;

13 **[0017]** FIGURE 4 is of a logic flow diagram illustrating creation of a virtual
14 development and/or test environment in one embodiment of the SYNONYM;

15 **[0018]** FIGURE 5 is of a logic flow diagram illustrating instantiation of user
16 accounts in one embodiment of the SYNONYM;

17 **[0019]** FIGURE 6A is of a block diagram illustrating copying of a synthetic
18 database from a "master" virtual development and/or test environment to new virtual
19 development and/or test environments in one embodiment of the SYNONYM;

20 **[0020]** FIGURE 6B is of a block diagram illustrating Developer Server Zone
21 (DvSZ) and Offshore Development Centers (ODC) in one embodiment of the
22 SYNONYM;

1 **[0021]** FIGURE 6C is of a block diagram illustrating dedicated use of hardware in
2 a DvSZ in one embodiment of the SYNONYM;

3 **[0022]** FIGURE 6D is of a block diagram illustrating physical access control at the
4 ODC in one embodiment of the SYNONYM;

5 **[0023]** FIGURE 6E is of a block diagram illustrating the use of public key
6 infrastructure (PKI) in one embodiment of the SYNONYM;

7 **[0024]** FIGURE 6F is of a block diagram illustrating transfer of sensitive info into
8 the DvSZ in one embodiment of the SYNONYM;

9 **[0025]** FIGURE 6G is of a block diagram illustrating DvSZ network layout in one
10 embodiment of the SYNONYM;

11 **[0026]** FIGURE 7 is of a block diagram illustrating embodiments of the
12 SYNONYM controller.

13 **[0027]** The leading number of each reference number within the drawings
14 indicates the figure in which that reference number is introduced and/or detailed. As
15 such, a detailed discussion of reference number 105 would be found and/or introduced
16 in Figure 1. Reference number 205 is introduced in Figure 2, etc.

17

DETAILED DESCRIPTION

OVERVIEW

[0028] As software applications have grown in size and complexity, testing of software applications has become increasingly more important. Having numerous components, including multiple database components, in large software applications exponentially increases the number of interactions that may happen among application components. Therefore, testing procedures should be used to check how changes in one application component affect other application components so as to prevent programming errors and inefficient implementations from being introduced into software applications as a result of changes to existing components or additions of new components. The amount and types of data used in software applications has also increased. As the number of users and options available to those users increases, the number of special cases, exceptions and languages supported by a software application also grows. Thus, the interactions between application components, including database components, and application data should also be tested to prevent errors. As the amount of testing that should be done increases, so does the cost, the length of time, and/or the number of people necessary to perform such testing. Some companies choose to outsource or offshore certain aspects of testing to reduce the cost associated with testing. However, outsourcing or offshoring testing creates another set of challenges as customer data useful in testing must be protected as required by various privacy laws and company policy. The synthetic deployment environment includes applications allowing mirroring by duplicating current environments, load balancing of

1 current systems by deploying a number of synthetic environments adding additional
2 capacity, generating synthetic gaming deployment environments, generating synthetic
3 educational deployment environments, generating synthetic training deployment
4 environment, generating synthetic medical deployment environments, fostering quality
5 improvement by allowing environmental variations for testing that does not affect
6 production environments.

7 SYNONYM

8 **[0029]** FIGURE 1A is of a block diagram illustrating the creation and use of a
9 virtual development and/or test environment in one embodiment of the SYNONYM. In
10 Figure 1A, a request to create a new virtual development and/or test environment is
11 received 105. In one embodiment, such a request may come from a member of a
12 development team and/or a testing team of a project, from a person responsible for
13 setting up development environments for projects, and/or the like. In another
14 embodiment, such a request may be issued automatically as part of creating a new
15 software application/component, modifying features of an application component,
16 adding support for new data types, and/or the like. The request may come in the form of
17 an email, a programmatic request to create a new virtual development and/or test
18 environment, and/or the like. In one implementation, a web based user interface may
19 provide text fields, checkboxes, radio buttons, and/or the like to accept user input
20 specifying administrative information and parameters indicative of the testing
21 requirements of a project. See FIGURE 1C for a screen shot diagram illustrating a web
22 based user interface in one embodiment of the SYNONYM. For example, such data may
23 include user name 181, user ID 182, list of required environments 183, project name

1 184, project ID 185, list of required applications 186, list of required databases 187,
 2 and/or the like. The parameters may be stored in XML, and upon submission by a user
 3 sent via HTTP post to a SYNONYM server. In one implementation, the request takes on
 4 the following form:

```

5  <XML>
6      <USER>
7          <USER NAME>Toni</USER NAME>
8          <USER ID>U1234</USER ID>
9      </USER>
10     <ENVIRONMENT>MAINFRAME</ENVIRONMENT>
11     <PROJECT>
12         <PROJECT NAME>New_Application</PROJECT NAME>
13         <PROJECT ID>P5678</PROJECT ID>
14     </PROJECT>
15     <REQUIRED APPLICATIONS>
16         <APPLICATION>Microsoft_Visio</APPLICATION>
17         <APPLICATION>Eclipse</APPLICATION>
18     </REQUIRED APPLICATIONS>
19     <REQUIRED DATABASES>
20         <DATABASE>SYN_CUSTOMERS</DATABASE>
21         <DATABASE>SYN_ACCOUNTS</DATABASE>
22         <DATABASE>SYN_DEPOSITS</DATABASE>
23     </REQUIRED DATABASES >
24 </XML>

```

25 **[0030]** In some cases, it may useful to limit access from actual data, for privacy
 26 concerns, policy concerns, and/or because US and international regulations limit the
 27 use and sharing of personal information and various regional, local, industry-wide, or
 28 company rules and/or customs limit the use and disclosure of information regarding
 29 certain business relationships or transactions of a person or entity (e.g., various
 30 restrictions may exist with regard to using real customer data for testing, especially

1 when such testing is outsourced or offshored). A virtual development and/or test
2 environment may therefore use synthesized data during testing instead of real data. In
3 one embodiment, such synthesized data may be based on anonymized real data. In one
4 implementation, anonymized data may be created 107 by replacing data values in
5 particular database fields with anonymized versions of those data values. Further details
6 regarding the creation of anonymized data are provided later in the specification. In one
7 implementation, synthesized data may be created 110 by synthesizing new data records
8 based on anonymized real data. In one implementation, the amount of data stored in a
9 synthetic database (synbase) may be reduced to increase efficiency. Further details
10 regarding the creation of synthesized data are provided later in the specification.

11 **[0031]** In alternative embodiments, real data and/or modified real data, such as
12 real data with modified IP addresses and/or modified domain/machine/server names,
13 may be used instead of synthesized data. In one embodiment, the synthetic deployment
14 environment includes applications allowing mirroring by duplicating current
15 environments. For example, real data and/or modified real data may be used for
16 mirroring using a synchronization technology, such as rsync, running through a job
17 scheduler, such as cron. In one embodiment, the IP addresses may be dynamically
18 assigned to such additional copies, the new domain/machine/server names may use the
19 previous name and add an incremental counter (e.g., PREVIOUS_NAME would become
20 PREVIOUS_NAME_02, PREVIOUS_NAME_03, etc.). In another embodiment, the
21 synthetic deployment environment includes applications allowing load balancing of
22 current systems by deploying a number of synthetic environments adding additional
23 capacity. For example, real data and/or modified real data may be used for load
24 balancing among multiple synthetic environments using a round robin scheduling

1 algorithm. In such an embodiment, the subsequently duplicated synthetic environments
2 with new identifiers may each become a part of the round robin pool geographically, or
3 based on the round-robin time allocation (e.g., each synthetic environment handling a
4 portion of work load for set quantum of time). In yet another embodiment, the synthetic
5 deployment environment includes applications allowing grid computing by deploying a
6 number of synthetic environments working in parallel. For example, real data and/or
7 modified real data may be used in grid computing using distributing computing
8 technology, such as Mac OS X Xgrid (e.g., where the duplicated renamed synthetic
9 environments act as individual processing nodes in the grid).

10 **[0032]** A virtual development and/or test environment may be created 115 in
11 response to the request to create a new virtual development and/or test environment. In
12 one embodiment, the virtual development and/or test environment may be persistent
13 (ENV) and shared across multiple projects. In another embodiment, the virtual
14 development and/or test environment may be created on demand (TED) and used by a
15 single project. In yet another embodiment, the virtual development and/or test
16 environment may use a database containing synthesized data records ("synbase"
17 hereinafter). In another implementation, the virtual development and/or test
18 environment may select one out of a plurality of synbase versions to use for testing. In
19 one embodiment, the virtual development and/or test environment may be physically
20 and/or logically separated from a production environment. See Figure 6C for a block
21 diagram illustrating dedicated use of hardware in a DvSZ in one embodiment of the
22 SYNONYM. In Figure 6C, no sharing of hardware is allowed between the DvSZ 620 and
23 the company intranet 621. An exception is made for the mainframe, and the mainframe

1 is instead separated logically into RZ1 622 and RZ8/9 623 between the company
2 intranet and the DvSZ respectively.

3 **[0033]** In one implementation, different security certificates (e.g., security
4 certificates based on Public Key Infrastructure) may be issued to resources in the virtual
5 development and/or test environment and in the production environment. See Figure
6 6E for an example SYNONYM embodiment showing a block diagram illustrating the use
7 of public key infrastructure (PKI). In Figure 6E, the PKI 640 in the company intranet
8 641 issues security certificates to computers in the company intranet 641 and in the
9 DvSZ 642. The PKI 640 issues different security certificates to resources in the company
10 intranet 641 and to resources in the DvSZ 642. A communication between a client and a
11 server can only succeed if both resources have either test security certificates or
12 production security certificates. For example, such resources may include Java
13 Application Platform servers, CORBA adapters, database servers, and/or the like. In one
14 implementation, a Development and Test Server Zone may be created to contain the
15 virtual development and/or test environment. For example, a dedicated network,
16 separated from the rest of an organization's intranet and limited in its ability to access
17 the organization's intranet, using components such as firewall, VPN Gateway, security
18 certificates, and/or the like may be setup using dedicated routers. Access to the DvSZ
19 may be limited to strongly authenticated users only and no sensitive information may be
20 accessible in the DvSZ. Furthermore, the DvSZ may be implemented as a "black hole"
21 network zone. As such, any information exchange between the DvSZ and outside data
22 sources is allowed only if it is initiated from outside the DvSZ. See Figure 6F for a block
23 diagram illustrating transfer of sensitive info into the DvSZ in one embodiment of the
24 SYNONYM. Figure 6F, illustrates that there is no automatic synchronization 650 of

1 information between repositories 651 in the company intranet 652 and repositories 653
2 in the DvSZ 654. Only sensitive information cleared through a quality gate 655A-B may
3 be transferred in the DvSZ. The establishment of a connection from inside the DvSZ 654
4 to a resource outside is blocked. Information cleared for use in the DvSZ may be
5 accessible only to those users of the DvSZ that are authorized to use the information.
6 See Figure 6G for a block diagram illustrating DvSZ network layout in one embodiment
7 of the SYNONYM. Figure 6G illustrates how the network is set up 660A-C, and the
8 hardware 661A-D used for the set up. Further details regarding the creation of a virtual
9 development and/or test environment are provided later in the specification. User
10 accounts authorized to use a virtual development and/or test environment may be
11 instantiated 120. In one embodiment, a variety of user account types, with a range of
12 rights and authentication requirements may be instantiated. Further details regarding
13 the instantiation of user accounts are provided later in the specification.

14 **[0034]** Authorized users may perform required development and testing 125
15 using a virtual development and/or testing environment and/or associated synbase. In
16 one embodiment, such testing may include unit testing, feature testing, application
17 testing, and/or the like. In one implementation, the testing may be performed by
18 dedicated human testers checking test results against specified performance
19 parameters. In another implementation, the testing may be performed by automated
20 test routines checking test results against predefined values or tested averages. For
21 example, the result of a "check account balance" operation used on a newly created
22 account may be compared against a predefined value of \$0 to ensure that the operation
23 behaves correctly for this test case. As another example, the time to complete "check
24 account balance" operations may be tracked and the average completion time compared

1 to the time taken to complete the current "check account balance" operation to test
2 execution speed. In another embodiment, randomized values may be placed into input
3 fields to determine if a program component handles errors properly, by logging/tracking
4 the program responses to such value entries. In another embodiment, application
5 source code may be tested for hard coding of sensitive data. In one implementation, a
6 Legal Parser may be used to identify potentially sensitive data within a source code file
7 written in a programming language, such as PL/I, Java, and/or the like, which might
8 violate established privacy or secrecy rules. For example, a department responsible for
9 specifying privacy rules, such as Legal and Compliance department, may specify a list of
10 potentially sensitive types of data (e.g., Names, CIF numbers, account numbers, and/or
11 the like). The Legal Parser may then be used to scan source code files of a project for
12 potential matches and provide notification of any such matches. For example, source
13 code and comments may be scanned using standard parsing techniques, such as by
14 using the m// operator in Perl, e.g., m/\$LegalParser/ for occurrences of a 12 digit
15 number, which corresponds to the length of a CIF number. In such an embodiment, a
16 LegalParser data structure (e.g., a file, memory struct, etc.) may house a list of flagged
17 terms, numbers, etc., which will be used to check for unauthorized use of such. A
18 variety of methods may be used to exchange information among test team members
19 during testing. In one implementation, email, instant messaging, http posts, and/or the
20 like protocols may be used to exchange information. For example, test team members
21 may send emails to each other using email software, such as Microsoft Outlook, Lotus
22 Notes, and/or the like, to ask questions and to receive answers. In another
23 implementation, technologies, such as Microsoft SharePoint, that provide document
24 management, discussion forums, surveys and email based notifications may be used to

1 exchange information. In one embodiment, the synthetic deployment environment may
2 be used to simulate a virtual bank. In another embodiment, the synthetic deployment
3 environment may be used for generating synthetic deployment environments in fields
4 such as, gaming, simulation, education, medicine, quality control, and/or the like.

5 **[0035]** A virtual development and/or test environment may be decommissioned
6 130 upon request. For example, a virtual development and/or test environment may be
7 decommissioned upon completion of testing required for a particular application
8 component, project phase, project, and/or the like. In one implementation, such
9 decommissioning may involve secure deletion, including multiple data wipes, of user
10 accounts and synbase associated with the virtual development and/or test environment.
11 In another embodiment, decommissioning may include archiving of the environment to
12 an owning company, and secure deletion of the environment at an outsourcing vendor
13 location.

14 **[0036]** FIGURE 1B is of a data flow diagram illustrating the transformation of real
15 data into synthetic data in one embodiment of the SYNONYM. In order to provide
16 realistic data for the virtual development and/or test environment, in one
17 implementation, real data 150 may be used as the basis for synthesized data. In one
18 implementation, real data may include fields such as customer name, customer address,
19 customer information file (CIF) number, and/or the like. In one embodiment, real data
20 is converted into anonymized data 160. In one implementation, anonymization may
21 involve replacing customer name and customer address fields with anonymized versions
22 of these fields created according to predefined rules. Further details regarding the
23 creation of anonymized data are provided later in the specification. In one embodiment,

1 anonymized data is converted into synthesized data 170. In one implementation, a
2 subset of anonymized data 165 representative of each customer type requested for
3 testing may be selected for synthesization. In one embodiment, the data set reduction is
4 responsive to more effectively test a particular type of data. For example, there may be
5 several types of accounts, SYNONYM may use just one or a select few types of accounts
6 for testing (e.g., premium accounts may hold a greater number and diversity of account
7 types and as such, may act as better data sets for stress testing a system). In another
8 embodiment, the subset acts to reduce the amount of synthesized data to facilitate faster
9 testing cycles. The selected data may be converted into synthesized data by creating a
10 new CIF for each selected customer and by updating the CIF number of each customer
11 to point to the newly created synthetic CIF. Further details regarding the creation of
12 synthesized data are provided later in the specification.

13 **[0037]** FIGURE 2A is of a logic flow diagram illustrating creation of anonymized
14 data in one embodiment of the SYNONYM. In Figure 2A, a request to anonymize data
15 records is received 205. In one embodiment, the request may contain a list of databases
16 251, database tables 252, and data fields 253 that need to be anonymized. The list of
17 databases, database tables, and data fields may be based on the input of a user who
18 requested a new test environment. In one implementation, the list may be in XML
19 format and the request may take on the following form:

```
20 <XML>
21     <ANONYMIZATION DATABASES>
22         <DATABASE>
23             <DATABASE NAME>REAL_CUSTOMERS</DATABASE NAME>
24             <TABLE>
25                 <TABLE NAME>Customer_Info</TABLE NAME>
```

```
1           <TABLE FIELD>Name</TABLE FIELD>
2           </TABLE>
3           </DATABASE>
4           </ANONYMIZATION DATABASES>
5 </XML>
```

6 **[0038]** In one embodiment, the above XML may result from user selections of
7 pop-up menu entries in a web form. See FIGURE 2C for a screen shot diagram
8 illustrating a web form based user interface in one embodiment of the SYNONYM. In
9 another implementation, the list may be stored in a database table and accessed by
10 SYNONYM via an SQL SELECT command. In yet another embodiment, the list may be
11 stored in a configuration file and accessed by SYNONYM using I/O calls that allow
12 reading from a file. Based on the list, the database tables to be anonymized are selected
13 207 and data records are extracted for processing 209. In one implementation, such
14 selection may be accomplished via one or more SQL SELECT commands (e.g., SELECT
15 CIF, Name FROM Customer_Info) and extraction may be accomplished by placing the
16 selected data records into a data file. Anonymization rules and reference tables may be
17 loaded at 213. Anonymization rules specify how to generate anonymized replacement
18 values for a particular field. FIGURE 2B is a table illustrating anonymization rules in
19 one embodiment of the SYNONYM. Reference tables contain data that may be used as
20 replacement values according to anonymization rules. For example, in one
21 implementation, a reference table may contain a list of 676 male and female names
22 along with a list of regions where each name is commonly used. In one embodiment, a
23 table is populated with made up names; in one embodiment a name generator (e.g.,
24 www.fakenamegenerator.com) may be used to generate and/or populate a database
25 table of made up names.

1 **[0039]** In one embodiment, a data record containing real data may be selected 215
2 from the data file for anonymization. For example, I/O calls that allow reading from a
3 data file may be used. The data record may be parsed to select the next available field
4 217 using a parsing algorithm. In one implementation, standard XML formatting may
5 be used to facilitate parsing. A determination is made whether the currently selected
6 field needs to be anonymized 219 based on the anonymization rules. For example, the
7 name of the field may be compared to the list of field names in the anonymization rules.
8 If the field name is not one of the field names specified in the anonymization rules, then
9 the current field value is kept 221. Otherwise, an appropriate rule for the field is selected
10 223 from the anonymization rules. In one implementation, the anonymization rules may
11 be loaded and comparison performed using a programming language, such as PL/I,
12 Perl, and/or the like. In one embodiment, an example of the PL/I rules may take the
13 following form:

```
14   if ($fieldType == TYPE_FIRST_NAME)
15       replaceFirstName($field);
16   if ($fieldType == TYPE_ADDRESS)
17       replaceAddress($field);
```

18
19 **[0040]** Based on the anonymization rule, an anonymized value is generated for
20 the field 225. In one embodiment, the anonymized value may be generated by randomly
21 selecting a value from a list of applicable values from a reference table specified in the
22 anonymization rules, e.g., from a made up name table. In another embodiment, the
23 anonymized value may be randomly generated based on instructions specified in the
24 anonymization rules. The generated anonymized value replaces the original field value
25 227. If there are more fields in the currently selected data record, the next field may be

1 selected for anonymization 229, otherwise, the next available record may be selected for
2 anonymization 231. Once all data records are anonymized, the anonymized data may be
3 output to one or more data files 233 using I/O calls that allow writing to a data file. In
4 one implementation, the data is written out to a new data file. In another
5 implementation, the existing data file is overwritten with new anonymized data. The
6 anonymized data is propagated 235 to allow all parts of SYNONYM access to the
7 anonymized data. For example, the anonymized data may be copied to files and/or
8 databases on storage partitions designed to work with various operating systems such as
9 Microsoft Windows XP, Unix, Apple Macintosh OS and/or the like.

10 **[0041]** FIGURE 3 is of a logic flow diagram illustrating creation of synthesized
11 data in one embodiment of the SYNONYM. In Figure 3, a request to create synthesized
12 data is received 310. In one embodiment, the request may contain a list of databases
13 that should be used in synthesization. The list of databases may be based on the input of
14 a user who requested a new test environment. In one implementation, the list may be in
15 XML format and the request may take on the following form:

```
16 <XML>
17     <SYNTHESIZATION DATABASES>
18         <DATABASE NAME>ANONYMIZED_CUSTOMERS</DATABASE NAME>
19         <DATABASE NAME>ANONYMIZED_ADDRESSES</DATABASE NAME>
20     </SYNTHESIZATION DATABASES>
21 </XML>
```

22 **[0042]** In another implementation, the list may be stored in a database table and
23 accessed by SYNONYM via SQL commands. For example, a web interface that allows a
24 user to select databases (e.g., via checkboxes) that should be used in the synthesization
25 may be used. The above XML may be sent to SYNONYM, which may connect to the

1 specified databases using database adapters, such as by using adapters available
2 through the Open Database Connectivity (ODBC) API protocol, adapters available
3 through the WebObjects Enterprise Objects Framework (EOF), and/or the like. In yet
4 another embodiment, the list may be stored in a configuration file and accessed by
5 SYNONYM using I/O calls that allow reading from a file. Based on the list, anonymized
6 data is input 315 for synthesization. In one implementation, the anonymized data is
7 input by providing the name of one or more files containing the anonymized data. In
8 another implementation, the anonymized data is input by providing the name of one or
9 more SQL databases and/or tables containing the anonymized data and/or through an
10 SQL SELECT of the specified tables. In yet another implementation, the anonymized
11 data is exported from one or more SQL database to one or more files and the names of
12 these files are input. In one embodiment, other data associated with the anonymized
13 data is also input.

14 **[0043]** The request to create synthesized data may also specify that a subset of the
15 data representative of each customer type requested for testing should be selected for
16 synthesization to reduce the amount of synthesized data. In one embodiment, the subset
17 of data that should be selected may be specified in terms of business constellations, and
18 business constellations may be specified in terms of equivalence classes. In one
19 implementation, an equivalence class may include: Title, Business Unit, Branch Office,
20 Nationality, Domicile, Language, Customer Type, Customer Segment, etc., and a
21 business constellation may include a unique combination of values of one or more
22 equivalence classes. For example, a business constellation may be specified as a male
23 business customer from the US. This business constellation may be passed in XML
24 format as part of the request as follows:

```
1  <XML>
2      <BUSINESS CONSTELLATIONS>
3          <BUSINESS CONSTELLATION>
4              <CUSTOMER TYPE>Male</CUSTOMER TYPE>
5              <CUSTOMER SEGMENT>Business</CUSTOMER SEGMENT>
6              <DOMICILE>US</DOMICILE>
7          </BUSINESS CONSTELLATION>
8      </BUSINESS CONSTELLATIONS>
9  </XML>
```

10 **[0044]** The data may be queried based on the specified business constellations
11 320. In one embodiment, programmatic commands, written in any programming
12 language, such as C++, Perl, SQL, and/or the like, may be issued to compare values of
13 the data to values specified in a business constellation. In another embodiment, an SQL
14 SELECT for a customer type may provide a subset of numerous search results, which
15 may be used. A subset of data representative of all specified business constellations may
16 then be obtained 330 based on the comparisons. In one implementation, the first data
17 record encountered that satisfies all equivalence classes of a given business constellation
18 is chosen to represent the given business constellation. Each business constellation may
19 also be represented by multiple data records up to some maximum specified number.

20 **[0045]** Synthesized data is based on the subset of data representative of all
21 specified business constellations. In one embodiment, new customer information files
22 with new CIF numbers may be created for each customer in the subset of data, and the
23 CIF numbers in the anonymized data and in other data associated with the anonymized
24 data may be replaced with new synthetic CIF numbers 335. For example, new CIF
25 numbers may be randomly and/or serially generated to be unique and to have the same
26 format as real CIF numbers. In one implementation, the number ranges used to

1 generate synthesized CIF numbers may be forbidden in the production environment.
2 After this is done, it is no longer possible to determine any actual customer
3 identification information from the new CIF numbers or to trace the new synthesized
4 CIF data to any actual customer. In one implementation, the resulting synthesized data
5 is output to one or more data files using I/O calls that allow writing to a data file.

6 **[0046]** A synthetic database may be created 345 to hold the new synthesized data.
7 In one embodiment, a new synbase is created using any number of database platforms
8 such as Microsoft SQL Server, MySQL, and/or the like. For example, it may be created
9 with the following mySQL SQL command:

```
10 create database DBname;
```

11

12 **[0047]** In another embodiment, contents of an existing synbase may be deleted
13 (e.g., using SQL DELETE FROM synbase_table_name [WHERE constraint]) and/or be
14 overwritten (e.g., using SQL UPDATE synbase_table_name SET
15 field_value='SomeNewValue' [WHERE constraint]) with synthesized data.

16 **[0048]** In an alternative embodiment, other data corresponding to the
17 synthesized data may be similarly synthesized before creating a synthetic database at
18 345. In one implementation, new accounts/deposits with new account/deposit numbers
19 may be created for accounts/deposits associated with the synthesized CIF data, and the
20 account/deposit numbers are replaced with new account/deposit numbers 340. For
21 example, new account/deposit numbers may be randomly generated to be unique and to
22 have the same format as real account/deposit numbers. In one implementation, the
23 number ranges used to generate synthesized account/deposit numbers may be
24 forbidden in the production environment.

1 **[0049]** The synthesized data is placed into a synbase 350. In one implementation,
2 the synthesized data, stored in one or more data files, may be read using I/O calls that
3 allow reading from a data file, parsed, such as by using the m// operator in Perl (e.g.,
4 m/|.*|/ to extract a field value enclosed by "|"), and placed into a synbase using SQL
5 commands. A determination is made as to whether the virtual development and/or test
6 environment should have transaction data in addition to synthesized data 355. In one
7 embodiment, such determination may be made by checking the request to synthesize
8 data for further information specifying end of day jobs (EODs) that should be run to
9 create transaction data. For example, in one implementation, this information may be
10 passed in XML format as part of the request as follows:

```
11 <XML>
12     <EODs>
13         <EOD>MakePayments</EOD>
14         <EOD>CancelPayments</EOD>
15     </EODs>
16 </XML>
```

17 **[0050]** In one embodiment, EODs may specify the selection criteria for
18 transactions that should be created for the virtual development and/or test environment
19 360. For example, in one implementation, an EOD may trigger the creation of a specific
20 type of transaction data (e.g., EOD MakePayments may create transaction data
21 simulating payments being made by customers, while EOD CancelPayments may create
22 transaction data simulating customers canceling payments). Thus, specifying that a
23 given EOD should be run to create transaction data, also determines the kind of
24 transaction data that should be created. Using EODs, a synbase version that includes
25 transaction data is created 365. In one embodiment, transaction data, such as money

1 transfers between two customers, may be simulated by an EOD by creating a specified
2 number of random money transfer transactions (e.g., in Perl to select two random
3 accounts, select a number within a specified range that represents an amount to be
4 transferred, and transfer that amount from the first to the second account), and
5 outputting data records representing these transactions to a file using I/O calls. In one
6 implementation the transaction data records along with synbase data are copied to a
7 new synbase (e.g., by using adapters available through the Open Database Connectivity
8 (ODBC) API standard, adapters available through the WebObjects Enterprise Objects
9 Framework (EOF), and/or the like), which comprises a new synbase version. In another
10 implementation only the transaction data records are copied into a separate database or
11 separate database table, and together with the synthesized data the two parts comprise a
12 new synbase version.

13 **[0051]** FIGURE 4 is of a logic flow diagram illustrating creation of a virtual
14 development and/or test environment in one embodiment of the SYNONYM. In Figure
15 4, a request to create a virtual development and/or test environment is received 440. In
16 one embodiment, this request may be based on the input of a user who requested a new
17 test environment, as described above, and may specify various environments,
18 applications and databases desired by the user. In one implementation, the request
19 takes on the following form:

```
20 <XML>
21     <ENVIRONMENT>MAINFRAME</ENVIRONMENT>
22     <REQUIRED APPLICATIONS>
23         <APPLICATION>Microsoft_Visio</APPLICATION>
24         <APPLICATION>Eclipse</APPLICATION>
25     </REQUIRED APPLICATIONS>
```

```
1.         <REQUIRED DATABASES>
2.             <DATABASE>SYN_CUSTOMERS</DATABASE>
3.             <DATABASE>SYN_ACCOUNTS</DATABASE>
4.             <DATABASE>SYN_DEPOSITS</DATABASE>
5.         </REQUIRED DATABASES >
6     </XML>
```

7 **[0052]** At 442, the requested runtime, middleware, database and/or the like
8 environments are installed. In one implementation, such software may comprise an
9 operating system, such as Microsoft Windows XP, Unix, Apple Macintosh OS and/or the
10 like; runtime environments, such as Java Runtime Environment, GTK+, and/ or the
11 like; middleware, such as CORBA, DCOM, RPC and/or the like; and database
12 environments, such as Microsoft SQL Server, MySQL, and/or the like. At 444, the
13 requested project specific applications, infrastructure and test tools are installed. In one
14 implementation, such software may comprise applications, such as a web server for
15 web-based projects; infrastructure, such as Concurrent Versioning System (CVS); and
16 test tools, such as SilkTest. The requested synbase version is selected 446 and copied to
17 one or more databases used by the virtual development and/or test environment 448. In
18 one embodiment, various synbase versions may reside on a "master" virtual
19 development and/or test environment. When a request to create a new virtual
20 development and/or test environment is received, the requested synbase version may be
21 copied from the "master" virtual development and/or test environment to the new
22 virtual development and/or test environment. In one implementation, a synbase version
23 with no transaction data may be copied from the "master" virtual development and/or
24 test environment to the new virtual development and/or test environment as part of the
25 FIT image. For example, such copying may be done by a fast copy utility such as IBM
26 FlashCopy. See Figure 6A for a block diagram illustrating copying of a synthetic

1 database from a "master" virtual development and/or test environment 601 to new
2 virtual development and/or test environments in one embodiment of the SYNONYM. In
3 Figure 6A, synbase version 1 is copied from the "master" virtual development and/or
4 test environment 601 to a persistent Env 1 virtual development and/or test environment
5 602 using FlashCopy, and application specific data described in 450 is added to the data
6 repository. Synbase version 3 is copied from the "master" virtual development and/or
7 test environment 601 to a new TED 1 virtual development and/or test environment 603
8 using FlashCopy, and new application specific data described in 450 is added to the data
9 repository.

10 **[0053]** In one embodiment, application specific data may be added 450 to the
11 data repository used by the virtual development and/or test environment. In one
12 implementation, Wiki pages containing project specific information may store their data
13 in one or more databases. In another implementation, source code in programming
14 languages such as Java, PL/I, and/or the like is added through the CVS.

15 **[0054]** FIGURE 5 is of a logic flow diagram illustrating instantiation of user
16 accounts in one embodiment of the SYNONYM. In Figure 5, a request to instantiate
17 users is received 540. In one embodiment, this request may be based on the input of a
18 user who requested a new test environment, as described above, and may specify a set of
19 users and account types associated with each user. In one implementation, account
20 types may be regular user, offshore user, and/or the like. Each account type may have
21 different authentication and authorization settings associated with it. The specified set
22 of users is granted access rights to the virtual development and/or test environment 542
23 and issued security certificates 544 to use for authentications and/or authorization. In

1 one implementation, such security certificates may be based on Public Key
2 Infrastructure and issued by a Certificate Authority. In another implementation, such
3 security certificates may be stored on SmartCards. In one embodiment, access to
4 resources may be limited to users that require those resources for their work. Thus,
5 users with the same account type may have access to different resources. In one
6 implementation, access rights may be revoked at well defined time points (e.g., at
7 project closure) or after a predefined time period (e.g., 6 months).

8 **[0055]** A First Installation Tool (FIT) image is a collection of software that may be
9 used to provide each computer in an organization with the same basic set of software. In
10 one embodiment a FIT image may contain an operating system, such as Windows XP; a
11 web browser, such as Microsoft Internet Explorer; an antivirus, such as McAfee
12 Antivirus; VPN software, such as CheckPoint VPN Connectivity and Integrity Client;
13 access software, such as IBM Personal Communications; and a set of office productivity
14 software, such as Microsoft Word, Microsoft Excel, Microsoft PowerPoint, and
15 Microsoft Outlook. In one implementation, development tools required by projects for
16 development and testing may exist as pre-packaged bundles for FIT (e.g., TIPD, SSH
17 clients, 3270 terminal emulation, Rational software, Eclipse, Clear Case, Clear Quest,
18 and/or the like). In one implementation, software included in a FIT image may differ
19 based on predefined profiles. For example, such profiles may include a profile for a
20 server, a desktop computer, a profile for a notebook computer, a profile for a mobile
21 device, and/or the like. In one implementation, the profiles may be specified as below:

```
22 <XML>  
23     <PROFILE>  
24         <TYPE>DESKTOP</TYPE>
```

```
1           <SOFTWARE>WindowsXP</SOFTWARE>
2           <SOFTWARE>MicrosoftOfficeXP</SOFTWARE>
3       </PROFILE>
4       <PROFILE>
5           <TYPE>MOBILE</TYPE>
6           <SOFTWARE>WindowsMobile6.0</SOFTWARE>
7           <SOFTWARE>MicrosoftOfficeMobile</SOFTWARE>
8       </PROFILE>
9 </XML>
```

10 **[0056]** Additional software may also be installed on a computer with a FIT image
11 on request. Such additional software may include programs such as Microsoft Visio,
12 Microsoft Project, Eclipse, and/or the like. At 546, a determination may be made which
13 users in the specified set of users are regular users and which are offshore users. In one
14 embodiment, users located locally may have a FIT image along with any additional
15 software installed on their computers 560 using a deployment tool, such as Windows
16 Deployment Services, and may access SYNONYM. In another embodiment, a VMware
17 disk image may be used to package and deploy a FIT image. For example, software
18 applications to be included in the FIT image may be packaged as VMware ThinApps and
19 included in the VMware disk image. Additional required applications may be supplied
20 with additional ThinApp installations over base (e.g., Desktop, Mobile, etc.)
21 installations. The VMware disk image may then be deployed to a workstation using
22 VMware software. In one embodiment, computers used by offshore users may be
23 located in a secure offshore development center (ODC), which may be hosted by a third
24 party provider. In one implementation, computers in an ODC are physically and/or
25 logically separated from any other computers and access to these computer is limited to
26 authorized personnel. For example, an ODC may be installed in physically separated

1 rooms with physical access control and dedicated assets and resources. The network in
2 the ODC may not be connected to the remaining network of the third party provider.
3 The ODC may be connected to the DvSZ via VPN and offshore users not working in the
4 ODC may not have access to the DvSZ. See Figure 6D for a block diagram illustrating
5 physical access control at the ODC in one embodiment of the SYNONYM. In figure 6D,
6 the ODC 630 is physically separated from the rest of the third party provider's site 631.
7 Only computers in the ODC 630 are allowed to connect with the DvSZ 632 and only
8 through a VPN tunnel 633.

9 **[0057]** Users located offshore may have a virtual FIT image along with any
10 additional software installed on their computers 548. In one implementation, software
11 on the offshore users' computers may include an operating system, such as Windows
12 XP; a virtualization software, such as VMware Player; and a FIT image used through the
13 virtualization software (virtual FIT image). In one implementation, the virtual FIT
14 image may be deployed using a Code Server located in the ODC. A Code Server may
15 serve to install FIT image on workstations in an ODC, to distribute additional software
16 to the workstations, and to provide updates (e.g., virus definition files updates for an
17 antivirus program) for software installed on the workstations. For example, the code
18 server may install an operating system, such as Windows XP, and a virtualization
19 software, such as VMware Player, on a workstation. VMware Player may then be used to
20 run a virtual FIT image packaged in a VMware disk image on the workstation. At 550, a
21 VPN Tunnel may be established from the ODC to the virtual development and/or test
22 environment in the DvSZ that allows offshore users access to SYNONYM. In one
23 implementation, authentication of the VPN counterparts may be done on a system level
24 and not on a user level. In one implementation, offshore users may have access only to

1 services inside the DvSZ. See Figure 6B for a block diagram illustrating Developer
2 Server Zone 610 and Offshore Development Centers 611A-C in one embodiment of the
3 SYNONYM. In Figure 6B, a DvSZ 610, separated from the rest of the company intranet
4 612, is connected to ODCs 1-3 through VPN tunnels 613A-C. The workstations inside the
5 ODCs 614A-F may communicate with services inside the DvSZ using the installed FIT
6 images obtained from code servers 615A-C. The workstations may not access services in
7 the company intranet outside the DvSZ.

8 SYNONYM Controller

9 **[0058]** FIGURE 7 illustrates inventive aspects of a SYNONYM controller 701 in a
10 block diagram. In this embodiment, the SYNONYM controller 701 may serve to
11 aggregate, process, store, search, serve, identify, instruct, generate, match, and/or
12 facilitate interactions with a computer through a variety of information technologies,
13 and/or other related data.

14 **[0059]** Typically, users, which may be people and/or other systems, may engage
15 information technology systems (e.g., computers) to facilitate information processing.
16 In turn, computers employ processors to process information; such processors 703 may
17 be referred to as central processing units (CPU). One form of processor is referred to as
18 a microprocessor. CPUs use communicative circuits to pass binary encoded signals
19 acting as instructions to enable various operations. These instructions may be
20 operational and/or data instructions containing and/or referencing other instructions
21 and data in various processor accessible and operable areas of memory 729 (e.g.,
22 registers, cache memory, random access memory, etc.). Such communicative
23 instructions may be stored and/or transmitted in batches (e.g., batches of instructions)

1 as programs and/or data components to facilitate desired operations. These stored
2 instruction codes, e.g., programs, may engage the CPU circuit components and other
3 motherboard and/or system components to perform desired operations. One type of
4 program is a computer operating system, which, may be executed by CPU on a
5 computer; the operating system enables and facilitates users to access and operate
6 computer information technology and resources. Some resources that may be employed
7 in information technology systems include: input and output mechanisms through
8 which data may pass into and out of a computer; memory storage into which data may
9 be saved; and processors by which information may be processed. These information
10 technology systems may be used to collect data for later retrieval, analysis, and
11 manipulation, which may be facilitated through a database program. These information
12 technology systems provide interfaces that allow users to access and operate various
13 system components.

14 **[0060]** In one embodiment, the SYNONYM controller 701 may be connected to
15 and/or communicate with entities such as, but not limited to: one or more users from
16 user input devices 711; peripheral devices 712; an optional cryptographic processor
17 device 728; and/or a communications network 713.

18 **[0061]** Networks are commonly thought to comprise the interconnection and
19 interoperation of clients, servers, and intermediary nodes in a graph topology. It should
20 be noted that the term “server” as used throughout this application refers generally to a
21 computer, other device, program, or combination thereof that processes and responds to
22 the requests of remote users across a communications network. Servers serve their
23 information to requesting “clients.” The term “client” as used herein refers generally to a

1 computer, program, other device, user and/or combination thereof that is capable of
2 processing and making requests and obtaining and processing any responses from
3 servers across a communications network. A computer, other device, program, or
4 combination thereof that facilitates, processes information and requests, and/or
5 furthers the passage of information from a source user to a destination user is
6 commonly referred to as a “node.” Networks are generally thought to facilitate the
7 transfer of information from source points to destinations. A node specifically tasked
8 with furthering the passage of information from a source to a destination is commonly
9 called a “router.” There are many forms of networks such as Local Area Networks
10 (LANs), Pico networks, Wide Area Networks (WANs), Wireless Networks (WLANs), etc.
11 For example, the Internet is generally accepted as being an interconnection of a
12 multitude of networks whereby remote clients and servers may access and interoperate
13 with one another.

14 **[0062]** The SYNONYM controller 701 may be based on computer systems that
15 may comprise, but are not limited to, components such as: a computer systemization
16 702 connected to memory 729.

17 Computer Systemization

18 **[0063]** A computer systemization 702 may comprise a clock 730, central
19 processing unit (“CPU(s)” and/or “processor(s)” (these terms are used interchangeable
20 throughout the disclosure unless noted to the contrary)) 703, a memory 729 (e.g., a read
21 only memory (ROM) 706, a random access memory (RAM) 705, etc.), and/or an
22 interface bus 707, and most frequently, although not necessarily, are all interconnected
23 and/or communicating through a system bus 704 on one or more (mother)board(s) 702

1 having conductive and/or otherwise transportive circuit pathways through which
2 instructions (e.g., binary encoded signals) may travel to effect communications,
3 operations, storage, etc. Optionally, the computer systemization may be connected to an
4 internal power source 786. Optionally, a cryptographic processor 726 may be connected
5 to the system bus. The system clock typically has a crystal oscillator and generates a base
6 signal through the computer systemization's circuit pathways. The clock is typically
7 coupled to the system bus and various clock multipliers that will increase or decrease
8 the base operating frequency for other components interconnected in the computer
9 systemization. The clock and various components in a computer systemization drive
10 signals embodying information throughout the system. Such transmission and reception
11 of instructions embodying information throughout a computer systemization may be
12 commonly referred to as communications. These communicative instructions may
13 further be transmitted, received, and the cause of return and/or reply communications
14 beyond the instant computer systemization to: communications networks, input
15 devices, other computer systemizations, peripheral devices, and/or the like. Of course,
16 any of the above components may be connected directly to one another, connected to
17 the CPU, and/or organized in numerous variations employed as exemplified by various
18 computer systems.

19 **[0064]** The CPU comprises at least one high-speed data processor adequate to
20 execute program components for executing user and/or system-generated requests.
21 Often, the processors themselves will incorporate various specialized processing units,
22 such as, but not limited to: integrated system (bus) controllers, memory management
23 control units, floating point units, and even specialized processing sub-units like
24 graphics processing units, digital signal processing units, and/or the like. Additionally,

1 processors may include internal fast access addressable memory, and be capable of
2 mapping and addressing memory 529 beyond the processor itself; internal memory may
3 include, but is not limited to: fast registers, various levels of cache memory (e.g., level 1,
4 2, 3, etc.), RAM, etc. The processor may access this memory through the use of a
5 memory address space that is accessible via instruction address, which the processor
6 can construct and decode allowing it to access a circuit path to a specific memory
7 address space having a memory state. The CPU may be a microprocessor such as:
8 AMD's Athlon, Duron and/or Opteron; ARM's application, embedded and secure
9 processors; IBM and/or Motorola's DragonBall and PowerPC; IBM's and Sony's Cell
10 processor; Intel's Celeron, Core (2) Duo, Itanium, Pentium, Xeon, and/or XScale;
11 and/or the like processor(s). The CPU interacts with memory through instruction
12 passing through conductive and/or transportive conduits (e.g., (printed) electronic
13 and/or optic circuits) to execute stored instructions (i.e., program code) according to
14 conventional data processing techniques. Such instruction passing facilitates
15 communication within the SYNONYM controller and beyond through various interfaces.
16 Should processing requirements dictate a greater amount speed and/or capacity,
17 distributed processors (e.g., Distributed SYNONYM), mainframe, multi-core, parallel,
18 and/or super-computer architectures may similarly be employed. Alternatively, should
19 deployment requirements dictate greater portability, smaller Personal Digital Assistants
20 (PDAs) may be employed.

21 **[0065]** Depending on the particular implementation, features of the SYNONYM
22 may be achieved by implementing a microcontroller such as CAST's R8051XC2
23 microcontroller; Intel's MCS 51 (i.e., 8051 microcontroller); and/or the like. Also, to
24 implement certain features of the SYNONYM, some feature implementations may rely

1 on embedded components, such as: Application-Specific Integrated Circuit ("ASIC"),
2 Digital Signal Processing ("DSP"), Field Programmable Gate Array ("FPGA"), and/or the
3 like embedded technology. For example, any of the SYNONYM component collection
4 (distributed or otherwise) and/or features may be implemented via the microprocessor
5 and/or via embedded components; e.g., via ASIC, coprocessor, DSP, FPGA, and/or the
6 like. Alternately, some implementations of the SYNONYM may be implemented with
7 embedded components that are configured and used to achieve a variety of features or
8 signal processing.

9 **[0066]** Depending on the particular implementation, the embedded components
10 may include software solutions, hardware solutions, and/or some combination of both
11 hardware/software solutions. For example, SYNONYM features discussed herein may
12 be achieved through implementing FPGAs, which are a semiconductor devices
13 containing programmable logic components called "logic blocks", and programmable
14 interconnects, such as the high performance FPGA Virtex series and/or the low cost
15 Spartan series manufactured by Xilinx. Logic blocks and interconnects can be
16 programmed by the customer or designer, after the FPGA is manufactured, to
17 implement any of the SYNONYM features. A hierarchy of programmable interconnects
18 allow logic blocks to be interconnected as needed by the SYNONYM system
19 designer/administrator, somewhat like a one-chip programmable breadboard. An
20 FPGA's logic blocks can be programmed to perform the function of basic logic gates
21 such as AND, and XOR, or more complex combinational functions such as decoders or
22 simple mathematical functions. In most FPGAs, the logic blocks also include memory
23 elements, which may be simple flip-flops or more complete blocks of memory. In some
24 circumstances, the SYNONYM may be developed on regular FPGAs and then migrated

1 into a fixed version that more resembles ASIC implementations. Alternate or
2 coordinating implementations may migrate SYNONYM controller features to a final
3 ASIC instead of or in addition to FPGAs. Depending on the implementation all of the
4 aforementioned embedded components and microprocessors may be considered the
5 “CPU” and/or “processor” for the SYNONYM.

6 **Power Source**

7 **[0067]** The power source 786 may be of any standard form for powering small
8 electronic circuit board devices such as the following power cells: alkaline, lithium
9 hydride, lithium ion, lithium polymer, nickel cadmium, solar cells, and/or the like.
10 Other types of AC or DC power sources may be used as well. In the case of solar cells, in
11 one embodiment, the case provides an aperture through which the solar cell may
12 capture photonic energy. The power cell 786 is connected to at least one of the
13 interconnected subsequent components of the SYNONYM thereby providing an electric
14 current to all subsequent components. In one example, the power source 786 is
15 connected to the system bus component 704. In an alternative embodiment, an outside
16 power source 786 is provided through a connection across the I/O 708 interface. For
17 example, a USB and/or IEEE 1394 connection carries both data and power across the
18 connection and is therefore a suitable source of power.

19 **Interface Adapters**

20 **[0068]** Interface bus(es) 707 may accept, connect, and/or communicate to a
21 number of interface adapters, conventionally although not necessarily in the form of
22 adapter cards, such as but not limited to: input output interfaces (I/O) 708, storage
23 interfaces 709, network interfaces 710, and/or the like. Optionally, cryptographic

1 processor interfaces 727 similarly may be connected to the interface bus. The interface
2 bus provides for the communications of interface adapters with one another as well as
3 with other components of the computer systemization. Interface adapters are adapted
4 for a compatible interface bus. Interface adapters conventionally connect to the
5 interface bus via a slot architecture. Conventional slot architectures may be employed,
6 such as, but not limited to: Accelerated Graphics Port (AGP), Card Bus, (Extended)
7 Industry Standard Architecture ((E)ISA), Micro Channel Architecture (MCA), NuBus,
8 Peripheral Component Interconnect (Extended) (PCI(X)), PCI Express, Personal
9 Computer Memory Card International Association (PCMCIA), and/or the like.

10 **[0069]** Storage interfaces 709 may accept, communicate, and/or connect to a
11 number of storage devices such as, but not limited to: storage devices 714, removable
12 disc devices, and/or the like. Storage interfaces may employ connection protocols such
13 as, but not limited to: (Ultra) (Serial) Advanced Technology Attachment (Packet
14 Interface) ((Ultra) (Serial) ATA(PI)), (Enhanced) Integrated Drive Electronics ((E)IDE),
15 Institute of Electrical and Electronics Engineers (IEEE) 1394, fiber channel, Small
16 Computer Systems Interface (SCSI), Universal Serial Bus (USB), and/or the like.

17 **[0070]** Network interfaces 710 may accept, communicate, and/or connect to a
18 communications network 713. Through a communications network 713, the SYNONYM
19 controller is accessible through remote clients 733b (e.g., computers with web browsers)
20 by users 733a. Network interfaces may employ connection protocols such as, but not
21 limited to: direct connect, Ethernet (thick, thin, twisted pair 10/100/1000 Base T,
22 and/or the like), Token Ring, wireless connection such as IEEE 802.11a-x, and/or the
23 like. Should processing requirements dictate a greater amount speed and/or capacity,

1 distributed network controllers (e.g., Distributed SYNONYM), architectures may
2 similarly be employed to pool, load balance, and/or otherwise increase the
3 communicative bandwidth required by the SYNONYM controller. A communications
4 network may be any one and/or the combination of the following: a direct
5 interconnection; the Internet; a Local Area Network (LAN); a Metropolitan Area
6 Network (MAN); an Operating Missions as Nodes on the Internet (OMNI); a secured
7 custom connection; a Wide Area Network (WAN); a wireless network (e.g., employing
8 protocols such as, but not limited to a Wireless Application Protocol (WAP), I-mode,
9 and/or the like); and/or the like. A network interface may be regarded as a specialized
10 form of an input output interface. Further, multiple network interfaces 710 may be used
11 to engage with various communications network types 713. For example, multiple
12 network interfaces may be employed to allow for the communication over broadcast,
13 multicast, and/or unicast networks.

14 **[0071]** Input Output interfaces (I/O) 708 may accept, communicate, and/or
15 connect to user input devices 711, peripheral devices 712, cryptographic processor
16 devices 728, and/or the like. I/O may employ connection protocols such as, but not
17 limited to: audio: analog, digital, monaural, RCA, stereo, and/or the like; data: Apple
18 Desktop Bus (ADB), IEEE 1394a-b, serial, universal serial bus (USB); infrared; joystick;
19 keyboard; midi; optical; PC AT; PS/2; parallel; radio; video interface: Apple Desktop
20 Connector (ADC), BNC, coaxial, component, composite, digital, Digital Visual Interface
21 (DVI), high-definition multimedia interface (HDMI), RCA, RF antennae, S-Video, VGA,
22 and/or the like; wireless: 802.11a/b/g/n/x, Bluetooth, code division multiple access
23 (CDMA), global system for mobile communications (GSM), WiMax, etc.; and/or the
24 like. One typical output device may include a video display, which typically comprises a

1 Cathode Ray Tube (CRT) or Liquid Crystal Display (LCD) based monitor with an
2 interface (e.g., DVI circuitry and cable) that accepts signals from a video interface, may
3 be used. The video interface composites information generated by a computer
4 systemization and generates video signals based on the composited information in a
5 video memory frame. Another output device is a television set, which accepts signals
6 from a video interface. Typically, the video interface provides the composited video
7 information through a video connection interface that accepts a video display interface
8 (e.g., an RCA composite video connector accepting an RCA composite video cable; a DVI
9 connector accepting a DVI display cable, etc.).

10 **[0072]** User input devices 711 may be card readers, dongles, finger print readers,
11 gloves, graphics tablets, joysticks, keyboards, mouse (mice), remote controls, retina
12 readers, trackballs, trackpads, and/or the like.

13 **[0073]** Peripheral devices 712 may be connected and/or communicate to I/O
14 and/or other facilities of the like such as network interfaces, storage interfaces, and/or
15 the like. Peripheral devices may be audio devices, cameras, dongles (e.g., for copy
16 protection, ensuring secure transactions with a digital signature, and/or the like),
17 external processors (for added functionality), goggles, microphones, monitors, network
18 interfaces, printers, scanners, storage devices, video devices, video sources, visors,
19 and/or the like.

20 **[0074]** It should be noted that although user input devices and peripheral devices
21 may be employed, the SYNONYM controller may be embodied as an embedded,
22 dedicated, and/or monitor-less (i.e., headless) device, wherein access would be provided
23 over a network interface connection.

1 **[0075]** Cryptographic units such as, but not limited to, microcontrollers,
2 processors 726, interfaces 727, and/or devices 728 may be attached, and/or
3 communicate with the SYNONYM controller. A MC68HC16 microcontroller,
4 manufactured by Motorola Inc., may be used for and/or within cryptographic units. The
5 MC68HC16 microcontroller utilizes a 16-bit multiply-and-accumulate instruction in the
6 16 MHz configuration and requires less than one second to perform a 512-bit RSA
7 private key operation. Cryptographic units support the authentication of
8 communications from interacting agents, as well as allowing for anonymous
9 transactions. Cryptographic units may also be configured as part of CPU. Equivalent
10 microcontrollers and/or processors may also be used. Other commercially available
11 specialized cryptographic processors include: the Broadcom's CryptoNetX and other
12 Security Processors; nCipher's nShield, SafeNet's Luna PCI (e.g., 7100) series;
13 Semaphore Communications' 40 MHz Roadrunner 184; Sun's Cryptographic
14 Accelerators (e.g., Accelerator 6000 PCIe Board, Accelerator 500 Daughtercard); Via
15 Nano Processor (e.g., L2100, L2200, U2400) line, which is capable of performing 500+
16 MB/s of cryptographic instructions; VLSI Technology's 33 MHz 6868; and/or the like.

17 **Memory**

18 **[0076]** Generally, any mechanization and/or embodiment allowing a processor to
19 affect the storage and/or retrieval of information is regarded as memory 729. However,
20 memory is a fungible technology and resource, thus, any number of memory
21 embodiments may be employed in lieu of or in concert with one another. It is to be
22 understood that the SYNONYM controller and/or a computer systemization may
23 employ various forms of memory 729. For example, a computer systemization may be

1 configured wherein the functionality of on-chip CPU memory (e.g., registers), RAM,
2 ROM, and any other storage devices are provided by a paper punch tape or paper punch
3 card mechanism; of course such an embodiment would result in an extremely slow rate
4 of operation. In a typical configuration, memory 729 will include ROM 706, RAM 705,
5 and a storage device 714. A storage device 714 may be any conventional computer
6 system storage. Storage devices may include a drum; a (fixed and/or removable)
7 magnetic disk drive; a magneto-optical drive; an optical drive (i.e., Blu-ray, CD
8 ROM/RAM/Recordable (R)/ReWritable (RW), DVD R/RW, HD DVD R/RW etc.); an
9 array of devices (e.g., Redundant Array of Independent Disks (RAID)); solid state
10 memory devices (USB memory, solid state drives (SSD), etc.); other processor-readable
11 storage mediums; and/or other devices of the like. Thus, a computer systemization
12 generally requires and makes use of memory.

13 **Component Collection**

14 **[0077]** The memory 729 may contain a collection of program and/or database
15 components and/or data such as, but not limited to: operating system component(s) 715
16 (operating system); information server component(s) 716 (information server); user
17 interface component(s) 717 (user interface); Web browser component(s) 718 (Web
18 browser); database(s) 719; mail server component(s) 721; mail client component(s) 722;
19 cryptographic server component(s) 720 (cryptographic server); the SYNONYM
20 component(s) 735; and/or the like (i.e., collectively a component collection). These
21 components may be stored and accessed from the storage devices and/or from storage
22 devices accessible through an interface bus. Although non-conventional program
23 components such as those in the component collection, typically, are stored in a local

1 storage device 714, they may also be loaded and/or stored in memory such as:
2 peripheral devices, RAM, remote storage facilities through a communications network,
3 ROM, various forms of memory, and/or the like.

4 **Operating System**

5 **[0078]** The operating system component 715 is an executable program component
6 facilitating the operation of the SYNONYM controller. Typically, the operating system
7 facilitates access of I/O, network interfaces, peripheral devices, storage devices, and/or
8 the like. The operating system may be a highly fault tolerant, scalable, and secure system
9 such as: Apple Macintosh OS X (Server); AT&T Plan 9; Be OS; Unix and Unix-like
10 system distributions (such as AT&T's UNIX; Berkley Software Distribution (BSD)
11 variations such as FreeBSD, NetBSD, OpenBSD, and/or the like; Linux distributions
12 such as Red Hat, Ubuntu, and/or the like); and/or the like operating systems. However,
13 more limited and/or less secure operating systems also may be employed such as Apple
14 Macintosh OS, IBM OS/2, Microsoft DOS, Microsoft Windows
15 2000/2003/3.1/95/98/CE/Millennium/NT/Vista/XP (Server), Palm OS, and/or the like.
16 An operating system may communicate to and/or with other components in a
17 component collection, including itself, and/or the like. Most frequently, the operating
18 system communicates with other program components, user interfaces, and/or the like.
19 For example, the operating system may contain, communicate, generate, obtain, and/or
20 provide program component, system, user, and/or data communications, requests,
21 and/or responses. The operating system, once executed by the CPU, may enable the
22 interaction with communications networks, data, I/O, peripheral devices, program
23 components, memory, user input devices, and/or the like. The operating system may

1 provide communications protocols that allow the SYNONYM controller to communicate
2 with other entities through a communications network 713. Various communication
3 protocols may be used by the SYNONYM controller as a subcarrier transport
4 mechanism for interaction, such as, but not limited to: multicast, TCP/IP, UDP, unicast,
5 and/or the like.

6 **Information Server**

7 **[0079]** An information server component 716 is a stored program component that
8 is executed by a CPU. The information server may be a conventional Internet
9 information server such as, but not limited to Apache Software Foundation's Apache,
10 Microsoft's Internet Information Server, and/or the like. The information server may
11 allow for the execution of program components through facilities such as Active Server
12 Page (ASP), ActiveX, (ANSI) (Objective-) C (++), C# and/or .NET, Common Gateway
13 Interface (CGI) scripts, dynamic (D) hypertext markup language (HTML), FLASH, Java,
14 JavaScript, Practical Extraction Report Language (PERL), Hypertext Pre-Processor
15 (PHP), pipes, Python, wireless application protocol (WAP), WebObjects, and/or the like.
16 The information server may support secure communications protocols such as, but not
17 limited to, File Transfer Protocol (FTP); HyperText Transfer Protocol (HTTP); Secure
18 Hypertext Transfer Protocol (HTTPS), Secure Socket Layer (SSL), messaging protocols
19 (e.g., America Online (AOL) Instant Messenger (AIM), Application Exchange (APEX),
20 ICQ, Internet Relay Chat (IRC), Microsoft Network (MSN) Messenger Service, Presence
21 and Instant Messaging Protocol (PRIM), Internet Engineering Task Force's (IETF's)
22 Session Initiation Protocol (SIP), SIP for Instant Messaging and Presence Leveraging
23 Extensions (SIMPLE), open XML-based Extensible Messaging and Presence Protocol

1 (XMPP) (i.e., Jabber or Open Mobile Alliance's (OMA's) Instant Messaging and
2 Presence Service (IMPS)), Yahoo! Instant Messenger Service, and/or the like. The
3 information server provides results in the form of Web pages to Web browsers, and
4 allows for the manipulated generation of the Web pages through interaction with other
5 program components. After a Domain Name System (DNS) resolution portion of an
6 HTTP request is resolved to a particular information server, the information server
7 resolves requests for information at specified locations on the SYNONYM controller
8 based on the remainder of the HTTP request. For example, a request such as
9 `http://123.124.125.126/myInformation.html` might have the IP portion of the request
10 "`123.124.125.126`" resolved by a DNS server to an information server at that IP address;
11 that information server might in turn further parse the http request for the
12 "`/myInformation.html`" portion of the request and resolve it to a location in memory
13 containing the information "`myInformation.html`." Additionally, other information
14 serving protocols may be employed across various ports, e.g., FTP communications
15 across port 21, and/or the like. An information server may communicate to and/or with
16 other components in a component collection, including itself, and/or facilities of the
17 like. Most frequently, the information server communicates with the SYNONYM
18 database 719, operating systems, other program components, user interfaces, Web
19 browsers, and/or the like.

20 **[0080]** Access to the SYNONYM database may be achieved through a number of
21 database bridge mechanisms such as through scripting languages as enumerated below
22 (e.g., CGI) and through inter-application communication channels as enumerated below
23 (e.g., CORBA, WebObjects, etc.). Any data requests through a Web browser are parsed
24 through the bridge mechanism into appropriate grammars as required by the

1 SYNONYM. In one embodiment, the information server would provide a Web form
2 accessible by a Web browser. Entries made into supplied fields in the Web form are
3 tagged as having been entered into the particular fields, and parsed as such. The entered
4 terms are then passed along with the field tags, which act to instruct the parser to
5 generate queries directed to appropriate tables and/or fields. In one embodiment, the
6 parser may generate queries in standard SQL by instantiating a search string with the
7 proper join/select commands based on the tagged text entries, wherein the resulting
8 command is provided over the bridge mechanism to the SYNONYM as a query. Upon
9 generating query results from the query, the results are passed over the bridge
10 mechanism, and may be parsed for formatting and generation of a new results Web page
11 by the bridge mechanism. Such a new results Web page is then provided to the
12 information server, which may supply it to the requesting Web browser.

13 **[0081]** Also, an information server may contain, communicate, generate, obtain,
14 and/or provide program component, system, user, and/or data communications,
15 requests, and/or responses.

16 User Interface

17 **[0082]** The function of computer interfaces in some respects is similar to
18 automobile operation interfaces. Automobile operation interface elements such as
19 steering wheels, gearshifts, and speedometers facilitate the access, operation, and
20 display of automobile resources, functionality, and status. Computer interaction
21 interface elements such as check boxes, cursors, menus, scrollers, and windows
22 (collectively and commonly referred to as widgets) similarly facilitate the access,
23 operation, and display of data and computer hardware and operating system resources,

1 functionality, and status. Operation interfaces are commonly called user interfaces.
2 Graphical user interfaces (GUIs) such as the Apple Macintosh Operating System's Aqua,
3 IBM's OS/2, Microsoft's Windows
4 2000/2003/3.1/95/98/CE/Millennium/NT/XP/Vista/7 (i.e., Aero), Unix's X-Windows
5 (e.g., which may include additional Unix graphic interface libraries and layers such as K
6 Desktop Environment (KDE), mythTV and GNU Network Object Model Environment
7 (GNOME)), web interface libraries (e.g., ActiveX, AJAX, (D)HTML, FLASH, Java,
8 JavaScript, etc. interface libraries such as, but not limited to, Dojo, jQuery(UI),
9 MooTools, Prototype, script.aculo.us, SWFObject, Yahoo! User Interface, any of which
10 may be used and) provide a baseline and means of accessing and displaying information
11 graphically to users.

12 **[0083]** A user interface component 717 is a stored program component that is
13 executed by a CPU. The user interface may be a conventional graphic user interface as
14 provided by, with, and/or atop operating systems and/or operating environments such
15 as already discussed. The user interface may allow for the display, execution,
16 interaction, manipulation, and/or operation of program components and/or system
17 facilities through textual and/or graphical facilities. The user interface provides a facility
18 through which users may affect, interact, and/or operate a computer system. A user
19 interface may communicate to and/or with other components in a component
20 collection, including itself, and/or facilities of the like. Most frequently, the user
21 interface communicates with operating systems, other program components, and/or the
22 like. The user interface may contain, communicate, generate, obtain, and/or provide
23 program component, system, user, and/or data communications, requests, and/or
24 responses.

Web Browser

1
2 **[0084]** A Web browser component 718 is a stored program component that is
3 executed by a CPU. The Web browser may be a conventional hypertext viewing
4 application such as Microsoft Internet Explorer or Netscape Navigator. Secure Web
5 browsing may be supplied with 128bit (or greater) encryption by way of HTTPS, SSL,
6 and/or the like. Web browsers allowing for the execution of program components
7 through facilities such as ActiveX, AJAX, (D)HTML, FLASH, Java, JavaScript, web
8 browser plug-in APIs (e.g., FireFox, Safari Plug-in, and/or the like APIs), and/or the
9 like. Web browsers and like information access tools may be integrated into PDAs,
10 cellular telephones, and/or other mobile devices. A Web browser may communicate to
11 and/or with other components in a component collection, including itself, and/or
12 facilities of the like. Most frequently, the Web browser communicates with information
13 servers, operating systems, integrated program components (e.g., plug-ins), and/or the
14 like; e.g., it may contain, communicate, generate, obtain, and/or provide program
15 component, system, user, and/or data communications, requests, and/or responses. Of
16 course, in place of a Web browser and information server, a combined application may
17 be developed to perform similar functions of both. The combined application would
18 similarly affect the obtaining and the provision of information to users, user agents,
19 and/or the like from the SYNONYM enabled nodes. The combined application may be
20 nugatory on systems employing standard Web browsers.

Mail Server

21
22 **[0085]** A mail server component 721 is a stored program component that is
23 executed by a CPU 703. The mail server may be a conventional Internet mail server such

1 as, but not limited to sendmail, Microsoft Exchange, and/or the like. The mail server
2 may allow for the execution of program components through facilities such as ASP,
3 ActiveX, (ANSI) (Objective-) C (++), C# and/or .NET, CGI scripts, Java, JavaScript,
4 PERL, PHP, pipes, Python, WebObjects, and/or the like. The mail server may support
5 communications protocols such as, but not limited to: Internet message access protocol
6 (IMAP), Messaging Application Programming Interface (MAPI)/Microsoft Exchange,
7 post office protocol (POP3), simple mail transfer protocol (SMTP), and/or the like. The
8 mail server can route, forward, and process incoming and outgoing mail messages that
9 have been sent, relayed and/or otherwise traversing through and/or to the SYNONYM.

10 **[0086]** Access to the SYNONYM mail may be achieved through a number of APIs
11 offered by the individual Web server components and/or the operating system.

12 **[0087]** Also, a mail server may contain, communicate, generate, obtain, and/or
13 provide program component, system, user, and/or data communications, requests,
14 information, and/or responses.

15 Mail Client

16 **[0088]** A mail client component 722 is a stored program component that is
17 executed by a CPU 703. The mail client may be a conventional mail viewing application
18 such as Apple Mail, Microsoft Entourage, Microsoft Outlook, Microsoft Outlook
19 Express, Mozilla, Thunderbird, and/or the like. Mail clients may support a number of
20 transfer protocols, such as: IMAP, Microsoft Exchange, POP3, SMTP, and/or the like. A
21 mail client may communicate to and/or with other components in a component
22 collection, including itself, and/or facilities of the like. Most frequently, the mail client
23 communicates with mail servers, operating systems, other mail clients, and/or the like;

1 e.g., it may contain, communicate, generate, obtain, and/or provide program
2 component, system, user, and/or data communications, requests, information, and/or
3 responses. Generally, the mail client provides a facility to compose and transmit
4 electronic mail messages.

Cryptographic Server

6 **[0089]** A cryptographic server component 720 is a stored program component
7 that is executed by a CPU 703, cryptographic processor 726, cryptographic processor
8 interface 727, cryptographic processor device 728, and/or the like. Cryptographic
9 processor interfaces will allow for expedition of encryption and/or decryption requests
10 by the cryptographic component; however, the cryptographic component, alternatively,
11 may run on a conventional CPU. The cryptographic component allows for the
12 encryption and/or decryption of provided data. The cryptographic component allows for
13 both symmetric and asymmetric (e.g., Pretty Good Protection (PGP)) encryption and/or
14 decryption. The cryptographic component may employ cryptographic techniques such
15 as, but not limited to: digital certificates (e.g., X.509 authentication framework), digital
16 signatures, dual signatures, enveloping, password access protection, public key
17 management, and/or the like. The cryptographic component will facilitate numerous
18 (encryption and/or decryption) security protocols such as, but not limited to: checksum,
19 Data Encryption Standard (DES), Elliptical Curve Encryption (ECC), International Data
20 Encryption Algorithm (IDEA), Message Digest 5 (MD5, which is a one way hash
21 function), passwords, Rivest Cipher (RC5), Rijndael, RSA (which is an Internet
22 encryption and authentication system that uses an algorithm developed in 1977 by Ron
23 Rivest, Adi Shamir, and Leonard Adleman), Secure Hash Algorithm (SHA), Secure

1 Socket Layer (SSL), Secure Hypertext Transfer Protocol (HTTPS), and/or the like.
2 Employing such encryption security protocols, the SYNONYM may encrypt all incoming
3 and/or outgoing communications and may serve as node within a virtual private
4 network (VPN) with a wider communications network. The cryptographic component
5 facilitates the process of "security authorization" whereby access to a resource is
6 inhibited by a security protocol wherein the cryptographic component effects authorized
7 access to the secured resource. In addition, the cryptographic component may provide
8 unique identifiers of content, e.g., employing and MD5 hash to obtain a unique
9 signature for an digital audio file. A cryptographic component may communicate to
10 and/or with other components in a component collection, including itself, and/or
11 facilities of the like. The cryptographic component supports encryption schemes
12 allowing for the secure transmission of information across a communications network
13 to enable the SYNONYM component to engage in secure transactions if so desired. The
14 cryptographic component facilitates the secure accessing of resources on the SYNONYM
15 and facilitates the access of secured resources on remote systems; i.e., it may act as a
16 client and/or server of secured resources. Most frequently, the cryptographic
17 component communicates with information servers, operating systems, other program
18 components, and/or the like. The cryptographic component may contain, communicate,
19 generate, obtain, and/or provide program component, system, user, and/or data
20 communications, requests, and/or responses.

21 The SYNONYM Database

22 **[0090]** The SYNONYM database component 719 may be embodied in a database
23 and its stored data. The database is a stored program component, which is executed by

1 the CPU; the stored program component portion configuring the CPU to process the
2 stored data. The database may be a conventional, fault tolerant, relational, scalable,
3 secure database such as Oracle or Sybase. Relational databases are an extension of a flat
4 file. Relational databases consist of a series of related tables. The tables are
5 interconnected via a key field. Use of the key field allows the combination of the tables
6 by indexing against the key field; i.e., the key fields act as dimensional pivot points for
7 combining information from various tables. Relationships generally identify links
8 maintained between tables by matching primary keys. Primary keys represent fields that
9 uniquely identify the rows of a table in a relational database. More precisely, they
10 uniquely identify rows of a table on the "one" side of a one-to-many relationship.

11 **[0091]** Alternatively, the SYNONYM database may be implemented using various
12 standard data-structures, such as an array, hash, (linked) list, struct, structured text file
13 (e.g., XML), table, and/or the like. Such data-structures may be stored in memory
14 and/or in (structured) files. In another alternative, an object-oriented database may be
15 used, such as Frontier, ObjectStore, Poet, Zope, and/or the like. Object databases can
16 include a number of object collections that are grouped and/or linked together by
17 common attributes; they may be related to other object collections by some common
18 attributes. Object-oriented databases perform similarly to relational databases with the
19 exception that objects are not just pieces of data but may have other types of
20 functionality encapsulated within a given object. If the SYNONYM database is
21 implemented as a data-structure, the use of the SYNONYM database 719 may be
22 integrated into another component such as the SYNONYM component 735. Also, the
23 database may be implemented as a mix of data structures, objects, and relational
24 structures. Databases may be consolidated and/or distributed in countless variations

1 through standard data processing techniques. Portions of databases, e.g., tables, may be
2 exported and/or imported and thus decentralized and/or integrated.

3 **[0092]** In one embodiment, the database component 719 includes several tables
4 719a-g. A user table 719a includes fields such as, but not limited to: customer ID, CIF
5 number, name, an address, and/or the like. The user table may support and/or track
6 multiple entity accounts on a SYNONYM. An account table 719b includes fields such as,
7 but not limited to: account number, account type, account opening date, asset ID, asset
8 value, and/or the like. A deposit table 719c includes fields such as, but not limited to:
9 deposit number, deposit type, deposit date, and/or the like. An anonymization table
10 719d includes fields such as, but not limited to: anonymization db name, anonymization
11 db id, anonymization db table, anonymization db field, and/or the like. A synthesization
12 table 719e includes fields such as, but not limited to: synthesization db name,
13 synthesization db id, synthesization db table, synthesization db field, and/or the like. A
14 transactions table 719f includes fields such as, but not limited to, transaction id,
15 transaction account one, transaction account two, transaction type, transaction amount,
16 and/or the like. An anonymization replacement data table 719g includes fields such as,
17 but not limited to, replacement first name, replacement last name, replacement address,
18 and/or the like.

19 **[0093]** In one embodiment, the SYNONYM database may interact with other
20 database systems. For example, employing a distributed database system, queries and
21 data access by search SYNONYM component may treat the combination of the
22 SYNONYM database, an integrated data security layer database as a single database
23 entity.

1 **[0094]** In one embodiment, user programs may contain various user interface
2 primitives, which may serve to update the SYNONYM. Also, various accounts may
3 require custom database tables depending upon the environments and the types of
4 clients the SYNONYM may need to serve. It should be noted that any unique fields may
5 be designated as a key field throughout. In an alternative embodiment, these tables have
6 been decentralized into their own databases and their respective database controllers
7 (i.e., individual database controllers for each of the above tables). Employing standard
8 data processing techniques, one may further distribute the databases over several
9 computer systemizations and/or storage devices. Similarly, configurations of the
10 decentralized database controllers may be varied by consolidating and/or distributing
11 the various database components 719a-g. The SYNONYM may be configured to keep
12 track of various settings, inputs, and parameters via database controllers.

13 **[0095]** The SYNONYM database may communicate to and/or with other
14 components in a component collection, including itself, and/or facilities of the like.
15 Most frequently, the SYNONYM database communicates with the SYNONYM
16 component, other program components, and/or the like. The database may contain,
17 retain, and provide information regarding other nodes and data.

18 The SYNONYMs

19 **[0096]** The SYNONYM component 735 is a stored program component that is
20 executed by a CPU. In one embodiment, the SYNONYM component incorporates any
21 and/or all combinations of the aspects of the SYNONYM that was discussed in the
22 previous figures. As such, the SYNONYM affects accessing, obtaining and the provision

1 of testing, information, services, transactions, and/or the like across various
2 communications networks.

3 **[0097]** The SYNONYM component enables the anonymization, synthesization,
4 testing of production environments, and/or the like and use of the SYNONYM.

5 **[0098]** The SYNONYM component enabling access of information between nodes
6 may be developed by employing standard development tools and languages such as, but
7 not limited to: Apache components, Assembly, ActiveX, binary executables, (ANSI)
8 (Objective-) C (++), C# and/or .NET, database adapters, CGI scripts, Java, JavaScript,
9 mapping tools, procedural and object oriented development tools, PERL, PHP, Python,
10 shell scripts, SQL commands, web application server extensions, web development
11 environments and libraries (e.g., Microsoft's ActiveX; Adobe AIR, FLEX & FLASH;
12 AJAX; (D)HTML; Dojo, Java; JavaScript; jQuery(UI); MooTools; Prototype;
13 script.aculo.us; Simple Object Access Protocol (SOAP); SWFObject; Yahoo! User
14 Interface; and/or the like), WebObjects, and/or the like. In one embodiment, the
15 SYNONYM server employs a cryptographic server to encrypt and decrypt
16 communications. The SYNONYM component may communicate to and/or with other
17 components in a component collection, including itself, and/or facilities of the like.
18 Most frequently, the SYNONYM component communicates with the SYNONYM
19 database, operating systems, other program components, and/or the like. The
20 SYNONYM may contain, communicate, generate, obtain, and/or provide program
21 component, system, user, and/or data communications, requests, and/or responses.

Distributed SYNONYMS

1
2 **[0099]** The structure and/or operation of any of the SYNONYM node controller
3 components may be combined, consolidated, and/or distributed in any number of ways
4 to facilitate virtual development and/or deployment. Similarly, the component
5 collection may be combined in any number of ways to facilitate deployment and/or
6 development. To accomplish this, one may integrate the components into a common
7 code base or in a facility that can dynamically load the components on demand in an
8 integrated fashion.

9 **[00100]** The component collection may be consolidated and/or distributed in
10 countless variations through standard data processing and/or development techniques.
11 Multiple instances of any one of the program components in the program component
12 collection may be instantiated on a single node, and/or across numerous nodes to
13 improve performance through load-balancing and/or data-processing techniques.
14 Furthermore, single instances may also be distributed across multiple controllers
15 and/or storage devices; e.g., databases. All program component instances and
16 controllers working in concert may do so through standard data processing
17 communication techniques.

18 **[00101]** The configuration of the SYNONYM controller will depend on the context
19 of system deployment. Factors such as, but not limited to, the budget, capacity, location,
20 and/or use of the underlying hardware resources may affect deployment requirements
21 and configuration. Regardless of if the configuration results in more consolidated
22 and/or integrated program components, results in a more distributed series of program
23 components, and/or results in some combination between a consolidated and

1 distributed configuration, data may be communicated, obtained, and/or provided.
2 Instances of components consolidated into a common code base from the program
3 component collection may communicate, obtain, and/or provide data. This may be
4 accomplished through intra-application data processing communication techniques
5 such as, but not limited to: data referencing (e.g., pointers), internal messaging, object
6 instance variable communication, shared memory space, variable passing, and/or the
7 like.

8 **[00102]** If component collection components are discrete, separate, and/or
9 external to one another, then communicating, obtaining, and/or providing data with
10 and/or to other component components may be accomplished through inter-application
11 data processing communication techniques such as, but not limited to: Application
12 Program Interfaces (API) information passage; (distributed) Component Object Model
13 ((D)COM), (Distributed) Object Linking and Embedding ((D)OLE), and/or the like),
14 Common Object Request Broker Architecture (CORBA), local and remote application
15 program interfaces Jini, Remote Method Invocation (RMI), SOAP, process pipes, shared
16 files, and/or the like. Messages sent between discrete component components for inter-
17 application communication or within memory spaces of a singular component for intra-
18 application communication may be facilitated through the creation and parsing of a
19 grammar. A grammar may be developed by using standard development tools such as
20 lex, yacc, XML, and/or the like, which allow for grammar generation and parsing
21 functionality, which in turn may form the basis of communication messages within and
22 between components. For example, a grammar may be arranged to recognize the tokens
23 of an HTTP post command, e.g.:

```
w3c -post http://... Value1
```

3 **[00103]** where Value1 is discerned as being a parameter because “http://” is part of
4 the grammar syntax, and what follows is considered part of the post value. Similarly,
5 with such a grammar, a variable “Value1” may be inserted into an “http://” post
6 command and then sent. The grammar syntax itself may be presented as structured data
7 that is interpreted and/or other wise used to generate the parsing mechanism (e.g., a
8 syntax description text file as processed by lex, yacc, etc.). Also, once the parsing
9 mechanism is generated and/or instantiated, it itself may process and/or parse
10 structured data such as, but not limited to: character (e.g., tab) delineated text, HTML,
11 structured text streams, XML, and/or the like structured data. In another embodiment,
12 inter-application data processing protocols themselves may have integrated and/or
13 readily available parsers (e.g., the SOAP parser) that may be employed to parse
14 communications data. Further, the parsing grammar may be used beyond message
15 parsing, but may also be used to parse: databases, data collections, data stores,
16 structured data, and/or the like. Again, the desired configuration will depend upon the
17 context, environment, and requirements of system deployment.

19 **[00104]** The entirety of this application (including the Cover Page, Title, Headings,
20 Field, Background, Summary, Brief Description of the Drawings, Detailed Description,
21 Claims, Abstract, Figures, and otherwise) shows by way of illustration various
22 embodiments in which the claimed inventions may be practiced. The advantages and
23 features of the application are of a representative sample of embodiments only, and are
24 not exhaustive and/or exclusive. They are presented only to assist in understanding and

1 teach the claimed principles. It should be understood that they are not representative of
2 all claimed inventions. As such, certain aspects of the disclosure have not been discussed
3 herein. That alternate embodiments may not have been presented for a specific portion
4 of the invention or that further undescribed alternate embodiments may be available for
5 a portion is not to be considered a disclaimer of those alternate embodiments. It will be
6 appreciated that many of those undescribed embodiments incorporate the same
7 principles of the invention and others are equivalent. Thus, it is to be understood that
8 other embodiments may be utilized and functional, logical, organizational, structural
9 and/or topological modifications may be made without departing from the scope and/or
10 spirit of the disclosure. As such, all examples and/or embodiments are deemed to be
11 non-limiting throughout this disclosure. Also, no inference should be drawn regarding
12 those embodiments discussed herein relative to those not discussed herein other than it
13 is as such for purposes of reducing space and repetition. For instance, it is to be
14 understood that the logical and/or topological structure of any combination of any
15 program components (a component collection), other components and/or any present
16 feature sets as described in the figures and/or throughout are not limited to a fixed
17 operating order and/or arrangement, but rather, any disclosed order is exemplary and
18 all equivalents, regardless of order, are contemplated by the disclosure. Furthermore, it
19 is to be understood that such features are not limited to serial execution, but rather, any
20 number of threads, processes, services, servers, and/or the like that may execute
21 asynchronously, concurrently, in parallel, simultaneously, synchronously, and/or the
22 like are contemplated by the disclosure. As such, some of these features may be
23 mutually contradictory, in that they cannot be simultaneously present in a single
24 embodiment. Similarly, some features are applicable to one aspect of the invention, and

1 inapplicable to others. In addition, the disclosure includes other inventions not
2 presently claimed. Applicant reserves all rights in those presently unclaimed inventions
3 including the right to claim such inventions, file additional applications, continuations,
4 continuations in part, divisions, and/or the like thereof. As such, it should be
5 understood that advantages, embodiments, examples, functional, features, logical,
6 organizational, structural, topological, and/or other aspects of the disclosure are not to
7 be considered limitations on the disclosure as defined by the claims or limitations on
8 equivalents to the claims.

9

CLAIMS

What is claimed is:

1. A data synthesizing processor-implemented method, comprising:

receiving a request to create synthesized testing data;

identifying real data records for anonymization;

selecting a source of conjured data for anonymization of real data records based

on a unique identifier in the real data records;

creating an anonymized data set by replacing real data in the real data records,

based on the unique identifier in the real data records, with the selected conjured data;

selecting a subset of the anonymized data set based on specified criteria for
synthesization;

generating a synthetic database, wherein generation further includes:

instantiating a shell synthetic database,

populating the shell synthetic data by copying the selected subset of the

anonymized data set into the shell synthetic database,

generating synthetic unique identifiers, and

replacing unique identifiers with synthetic unique identifiers; and

instantiating the synthetic database.

2. The method of claim 1, wherein, the specified criteria is provided by a user

including constraints based on a business constellation.

3. A data synthesizing apparatus, comprising:

a memory;

1 a processor disposed in communication with said memory, and configured to
2 issue a plurality of processing instructions stored in the memory, wherein the processor
3 issues instructions to:

4 receive a request to create synthesized testing data;
5 identify real data records for anonymization;
6 select a source of conjured data for anonymization of real data records
7 based on a unique identifier in the real data records;

8 create an anonymized data set by replacing real data in the real data
9 records, based on the unique identifier in the real data records, with the selected
10 conjured data;

11 select a subset of the anonymized data set based on specified criteria for
12 synthesization;

13 generate a synthetic database, wherein generation further includes
14 instructions to:

15 instantiate a shell synthetic database,
16 populate the shell synthetic data by copying the selected subset of
17 the anonymized data set into the shell synthetic database,
18 generate synthetic unique identifiers, and
19 replace unique identifiers with synthetic unique identifiers; and
20 instantiate the synthetic database.

21 4. A data synthesizing processor-readable medium storing processor-issuable
22 instructions to:

23 receive a request to create synthesized testing data;
24 identify real data records for anonymization;

1 select a source of conjured data for anonymization of real data records based on a
2 unique identifier in the real data records;

3 create an anonymized data set by replacing real data in the real data records,
4 based on the unique identifier in the real data records, with the selected conjured data;

5 select a subset of the anonymized data set based on specified criteria for
6 synthesization;

7 generate a synthetic database, wherein generation further includes instructions
8 to:

9 instantiate a shell synthetic database,

10 populate the shell synthetic data by copying the selected subset of the
11 anonymized data set into the shell synthetic database,

12 generate synthetic unique identifiers, and

13 replace unique identifiers with synthetic unique identifiers; and

14 instantiate the synthetic database.

15 5. A data synthesizing system, comprising means to:

16 means to receive a request to create synthesized testing data;

17 means to identify real data records for anonymization;

18 means to select a source of conjured data for anonymization of real data records
19 based on a unique identifier in the real data records;

20 means to create an anonymized data set by replacing real data in the real data
21 records, based on the unique identifier in the real data records, with the selected
22 conjured data;

23 means to select a subset of the anonymized data set based on specified criteria for
24 synthesization;

1 means to generate a synthetic database, wherein generation further includes:
2 means to instantiate a shell synthetic database,
3 means to populate the shell synthetic data by copying the selected subset
4 of the anonymized data set into the shell synthetic database,
5 means to generate synthetic unique identifiers, and
6 means to replace unique identifiers with synthetic unique identifiers; and
7 means to instantiate the synthetic database.

8 6. An environment testing processor-implemented method, comprising:
9 receiving a request to create a test environment;
10 identifying real data records based on the request;
11 generating anonymized data from the real data;
12 selecting a subset of the anonymized data based on the request;
13 generating synthesized data from the subset of anonymized data;
14 generating a synthetic test environment, wherein generation further includes:
15 instantiating user accounts,
16 populating the test environment with a profile of testing components,
17 loading the synthesized data into the test environment; and
18 deploying the synthetic test environment to a location specified in the request.

19 7. The method of claim 6, wherein, the request includes constraint criteria
20 for selecting a subset of data from the anonymized data for use in generating the
21 synthesized data.

22 8. The method of claim 6, further, comprising:
23 effecting performance of testing in the deployed synthetic test environment.

24 9. The method of claim 6, further, comprising:

1 effecting decommissioning of the deployed synthetic test environment.

2 10. An environment testing apparatus, comprising:

3 a memory;

4 a processor disposed in communication with said memory, and configured to
5 issue a plurality of processing instructions stored in the memory, wherein the processor
6 issues instructions to:

7 receive a request to create a test environment;

8 identify real data records based on the request;

9 generate anonymized data from the real data;

10 select a subset of the anonymized data based on the request;

11 generate synthesized data from the subset of anonymized data;

12 generate a synthetic test environment, wherein generation further includes

13 instructions to:

14 instantiate user accounts,

15 populate the test environment with a profile of testing components,

16 load the synthesized data into the test environment; and

17 deploy the synthetic test environment to a location specified in the
18 request.

19 11. An environment testing processor-readable medium storing processor-
20 issuable instructions to:

21 receive a request to create a test environment;

22 identify real data records based on the request;

23 generate anonymized data from the real data;

24 select a subset of the anonymized data based on the request;

1 generate synthesized data from the subset of anonymized data;
2 generate a synthetic test environment, wherein generation further includes
3 instructions to:

4 instantiate user accounts,
5 populate the test environment with a profile of testing components,
6 load the synthesized data into the test environment; and
7 deploy the synthetic test environment to a location specified in the request.

8 12. An environment testing system, comprising means to:

9 means to receive a request to create a test environment;
10 means to identify real data records based on the request;
11 means to generate anonymized data from the real data;
12 means to select a subset of the anonymized data based on the request;
13 means to generate synthesized data from the subset of anonymized data;
14 means to generate a synthetic test environment, wherein generation further
15 includes:

16 means to instantiate user accounts,
17 means to populate the test environment with a profile of testing
18 components,
19 means to load the synthesized data into the test environment; and
20 means to deploy the synthetic test environment to a location specified in the
21 request.

22 13. An environment deployment processor-implemented method, comprising:

23 receiving a request to create a deployment environment;
24 identifying real data records based on the request;

1 generating varianced data from the real data;
2 generating a synthetic deployment environment, wherein generation further
3 includes:
4 instantiating user accounts,
5 populating the synthetic deployment environment with a profile of
6 deployment components,
7 loading the varianced data into the synthetic deployment environment;
8 and
9 deploying the synthetic deployment environment to a location specified in the
10 request.

11 14. The method of claim 13, wherein the deployment environment is used to
12 create a virtual bank.

13 15. The method of claim 13, wherein the deployment environment is used for
14 generating synthetic gaming deployment environments.

15 16. The method of claim 13, wherein the deployment environment is used for
16 generating synthetic simulation deployment environments.

17 17. The method of claim 13, wherein the deployment environment is used for
18 generating synthetic educational deployment environments.

19 18. The method of claim 13, wherein the deployment environment is used for
20 generating synthetic medical deployment environments.

21 19. The method of claim 13, wherein the deployment environment is used for
22 quality improvement.

23 20. An environment deployment apparatus, comprising:
24 a memory;

1 a processor disposed in communication with said memory, and configured to
2 issue a plurality of processing instructions stored in the memory, wherein the processor
3 issues instructions to:

4 receive a request to create a deployment environment;
5 identify real data records based on the request;
6 generate varianced data from the real data;
7 generate a synthetic deployment environment, wherein generation further
8 includes instructions to:

9 instantiate user accounts,
10 populate the synthetic deployment environment with a profile of
11 deployment components,
12 load the varianced data into the synthetic deployment environment;

13 and

14 deploy the synthetic deployment environment to a location specified in the
15 request.

16 21. An environment deployment processor-readable medium storing
17 processor-issuable instructions to:

18 receive a request to create a deployment environment;
19 identify real data records based on the request;
20 generate varianced data from the real data;
21 generate a synthetic deployment environment, wherein generation further
22 includes instructions to:

23 instantiate user accounts,

1 populate the synthetic deployment environment with a profile of
2 deployment components,

3 load the varianced data into the synthetic deployment environment; and

4 deploy the synthetic deployment environment to a location specified in the
5 request.

6 22. An environment deployment system, comprising means to:

7 means to receive a request to create a deployment environment;

8 means to identify real data records based on the request;

9 means to generate varianced data from the real data;

10 means to generate a synthetic deployment environment, wherein generation
11 further includes:

12 means to instantiate user accounts,

13 means to populate the synthetic deployment environment with a profile of
14 deployment components,

15 means to load the varianced data into the synthetic deployment
16 environment; and

17 means to deploy the synthetic deployment environment to a location specified in
18 the request.

19 23. A processor-implemented method for synthesizing data, comprising:

20 receiving a request to create synthesized customer records;

21 identifying real data records for anonymization;

22 selecting a source of conjured data for the anonymization of real data
23 records;

24 replacing real data in the real data records with the selected conjured data;

1 retrieving anonymized customer records to be synthesized;
2 creating synthetic customer records with synthetic customer record
3 identifiers to hold anonymized data contained in the anonymized customer records;
4 copying the anonymized data into the synthetic customer records;
5 locating, via a processor, in the copied anonymized data customer record
6 identifiers;
7 replacing the located customer record identifiers with the synthetic
8 customer record identifiers; and
9 outputting the synthesized customer records.

10 24. The method of claim 23, wherein the anonymized data contained in the
11 retrieved anonymized customer records is created by:

12 receiving a request to create anonymized customer records;
13 retrieving real customer records to be anonymized;
14 locating, via a processor, in real data contained in the real customer
15 records sensitive customer information;
16 choosing anonymized customer information to replace the located
17 sensitive customer information according to predefined rules;
18 replacing the located sensitive customer information with the anonymized
19 customer information; and
20 outputting the anonymized customer records.

21 25. The method of claim 24, wherein the sensitive customer information is
22 one of: a name, an address, a phone number, a fax number, a date, a free text field, a
23 personal identification number.

24 26. The method of claim 23, further comprising:

1 associating anonymized data contained in an anonymized customer record
2 with a business constellation, wherein the business constellation is defined by a plurality
3 of equivalence classes; and

4 creating a synthetic customer record for the anonymized data contained in
5 the anonymized customer record only if the number of previously created synthetic
6 customer records containing anonymized data associated with the business
7 constellation is less than a predefined number.

8 27. The method of claim 26, wherein an equivalence class is one of: Title,
9 Business Unit, Branch Office, Nationality, Domicile, Language, Customer Type,
10 Customer Segment.

11 28. The method of claim 23, further comprising:

12 retrieving real account records to be synthesized;

13 creating synthetic account records with synthetic account record
14 identifiers to hold account data contained in the real account records;

15 copying the account data into the synthetic account records;

16 locating in the copied account data customer record identifiers;

17 replacing the located customer record identifiers with the synthetic
18 customer record identifiers;

19 locating in the copied account data account record identifiers;

20 replacing the located account record identifiers with the synthetic account
21 record identifiers; and

22 outputting the synthesized account records.

23 29. The method of claim 23, further comprising:

24 retrieving real deposit records to be synthesized;

1 creating synthetic deposit records with synthetic deposit record identifiers
2 to hold deposit data contained in the real deposit records;
3 copying the deposit data into the synthetic deposit records;
4 locating in the copied deposit data customer record identifiers;
5 replacing the located customer record identifiers with the synthetic
6 customer record identifiers;
7 locating in the copied deposit data deposit record identifiers;
8 replacing the located deposit record identifiers with the synthetic deposit
9 record identifiers; and
10 outputting the synthesized deposit records.

11 30. The method of claim 23, further comprising:
12 simulating transactions using the synthesized customer records; and
13 outputting data representative of the simulated transactions.

14 31. A system for facilitating development and testing of software applications
15 using synthesized customer records comprising:

16 a server;
17 a software application for development and testing;
18 a development environment running on the server, wherein the
19 development environment is generated by cloning portions of a production environment
20 that are required to run the software application;
21 a database containing synthesized customer records, wherein:
22 the database is communicatively connected with the development
23 environment; and

1 the database provides a source of data for the development
2 environment.

3 32. The system of claim 31, further comprising a Development and Test Server
4 Zone containing the system and separating the development environment from the
5 production environment.

6 33. The system of claim 31, wherein the development environment is
7 physically separated from the production environment.

8 34. The system of claim 31, wherein the development environment is logically
9 separated from the production environment.

10 35. The system of claim 31, wherein the development environment is
11 decommissioned upon completion of testing.

12 36. A method for development and testing of software applications using
13 synthesized customer records comprising:

14 connecting to a development environment with a workstation;

15 providing authentication information required to access the development
16 environment; and

17 testing, on the workstation, a software application, wherein testing the
18 software application involves accessing synthesized customer records.

19 37. The method of claim 36, further comprising installing a FIT image onto
20 the workstation to allow the workstation to connect to the development environment.

21 38. The method of claim 37, wherein the FIT image is installed in a virtual
22 machine installed on the workstation.

23 39. The method of claim 36, further comprising:

24 placing the workstation in an offshore development center; and

1 establishing a virtual private network tunnel between the offshore
2 development center and the development environment to connect to the development
3 environment.

4 40. The method of claim 36, wherein security certificates are used to provide
5 authentication information.

6 41. The method of claim 36, wherein SmartCards are used to provide
7 authentication information.

8 42. The method of claim 36, wherein only the synthesized customer records
9 required for testing may be accessed.

10

11