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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
25 August 2005 (25.08.2005)

PCT

(10) International Publication Number
WO 2005/078576 A1

(51) International Patent Classification[7]: G06F 9/44

(21) International Application Number:
PCT/CA2005/000054

(22) International Filing Date: 18 January 2005 (18.01.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/781,229 18 February 2004 (18.02.2004) US

(71) Applicant *(for all designated States except US)*: **ROBOCODER CORPORATION** [CA/CA]; 1075 West Georgia St., Suite 1380, Vancouver, British Columbia V6E 3C9 (CA).

(72) Inventor: **LIN, Nelson, H.**; 3471 Johnson Ave., Richmond, British Columbia V7C 5R2 (CA).

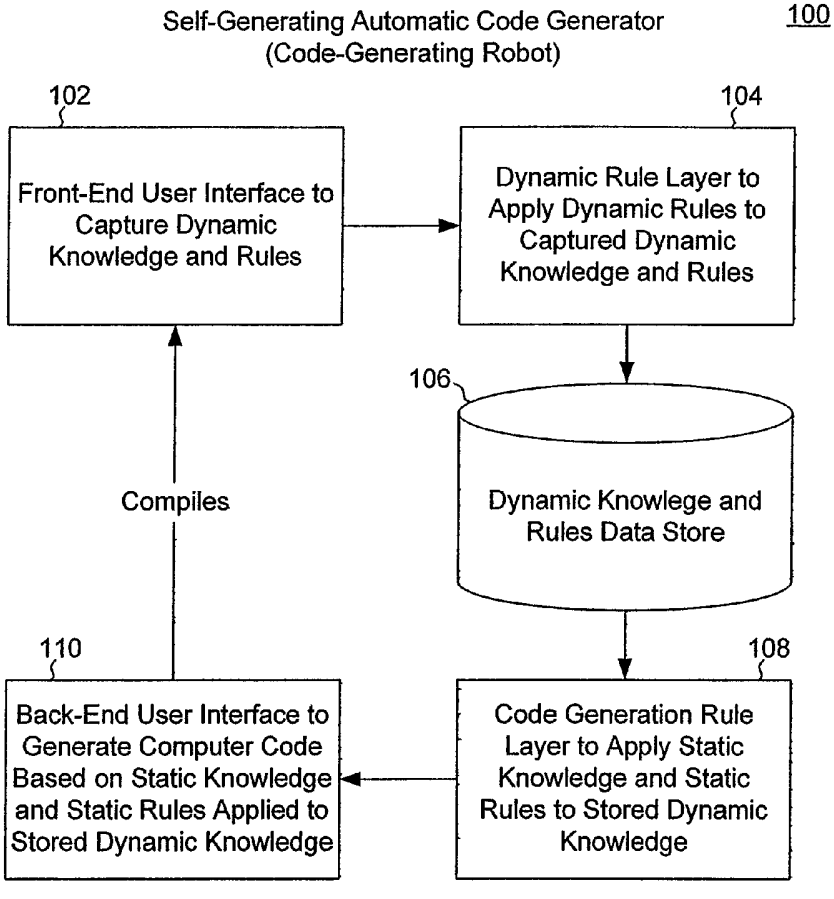
(74) Agent: **MACLEAN, Scott**; Cassan Maclean, Suite 401, 80 Aberdeen Street, Ottawa, Ontario K1S 5R5 (CA).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO,

[Continued on next page]

(54) Title: CODE GENERATOR FOR ADDING A NEW FEATURE



(57) Abstract: A self-generating automatic code generation system includes a user interface for capturing a set of knowledge and rules that define an application; a first automatic code generator for generating code for the application using the set of knowledge and rules; a user-interface for capturing a set of knowledge and rules that define a new feature to be added to the application; and a second automatic code generator supporting the new feature that was automatically generated by the first automatic code generator using the set of knowledge and rules defining the new feature as well as a set of knowledge and rules defining the first automatic code generator, wherein the second automatic code generator is to regenerate the code for the application such that the application includes the new feature using a set of knowledge and rules for generating the application as well as a set of knowledge and rules for using the new feature.

WO 2005/078576 A1

SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)*

Published:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

CODE GENERATOR FOR ADDING A NEW FEATURE

Technical Field

The present invention relates generally to the field of software development and, more particularly, to improvements in automatic code generation for enterprise software.

Background of the Invention

Reducing development and maintenance time is an increasingly important goal for developers of enterprise software. Automatic code generation is one approach to achieving this goal. Enterprise software applications are usually comprised of millions of lines of computer code and have traditionally required teams of programmers several years to create, test and debug. Today, an automatic code generator may be used to capture knowledge and business rules for an enterprise software application and generate the millions of lines of codes within seconds in any desired computer language and for any platform. An example of an automatic code generator includes GeneXus, a product of ARTech Consultores S.R.L.

Ironically, automatic code generators, which are themselves enterprise-level software applications, are developed from scratch using standard development tools and programming techniques, not using automatic code generation. This means that all the benefits attributable to automatic code generators do not apply to the development process of automatic code generators.

Another deficiency of conventional automatic code generators is the discontinuity that inevitably results between development and maintenance of enterprise software. Traditionally, an automatic code generator produces computer code for a target platform based on the captured knowledge and business rules. Once the computer code is generated, programmers customize the code to fix bugs, add new features, and the like. This breaks the continuity between the enterprise software and the automatic code generator. Subsequent development or maintenance on the enterprise software must be done the traditional way without the benefit of the automatic code generator.

The discontinuity problem is exacerbated where an automatic code generator is used to create several different software applications for an enterprise, all of which are subsequently modified or updated in various ways outside of the context of the automatic code generator. If a new feature is later desired for all of the software applications,

programmers must independently code the new feature into each application, thereby increasing development time and the possibility of bugs and incompatibility between applications.

What is needed is a way to bring all the benefits provided by an automatic code generator to the automatic code generator, itself, so that any enterprise software applications generated thereby can always rely on the automatic code generator to regenerate any computer code that these applications require. Furthermore, what is needed is a way for programmers to add a new feature only once and have the code generator automatically supply that feature to all of the other applications.

Brief Description of the Drawings

Non-exhaustive embodiments of the invention are now described with reference to the figures, in which

FIG. 1 is a block diagram of a self-generating automatic code generator, also referred to herein as a code-generating robot;

FIG. 2 is a user interface screen for capturing some of the dynamic knowledge required for generating itself or a different screen;

FIG. 3 is a user interface screen for capturing some of the dynamic rules required for the validation and organization of dynamic knowledge captured by the user interface screen of FIG. 2;

FIG. 4 is an entity relationship diagram showing some of the data stored and their relationships for the dynamic knowledge captured in FIG. 2;

FIG. 5 illustrates some of the static knowledge and rules required for the generation of computer code for a plurality of applications;

FIG. 6 is a user interface for the generation of computer codes based on static knowledge and static rules applied to captured dynamic knowledge;

FIG. 7 illustrates a process of seeding a first iteration of the code-generating robot;

FIG. 8 illustrates a process of enhancing the first iteration of the code-generating robot;

FIG. 9 is an example screen of an application generated with the new feature "Screen Filters" with the default filter radio button checked;

FIG. 10 is an example screen of an application generated with the new feature "Screen Filters" with the other filter radio button checked;

FIG. 11 is a table with columns used for the storage of dynamic knowledge for the new feature "Screen Filters" to appear in the desired applications;

FIG. 12 illustrates dynamic knowledge captured for the generation of the screen required to capture dynamic knowledge required for the new feature "Screen Filters";

FIG. 13 is a dialog for generating the screen required to capture dynamic knowledge for the new feature "Screen Filters";

FIG. 14 illustrates the screen to capture dynamic knowledge required for the new feature "Screen Filters";

FIG. 15 shows some of the code representing static knowledge and static rules for the generation of computer code needed for the new feature "Screen Filters";

FIG. 16 illustrates the fact that a screen used to generate the new feature also enjoy the same benefit of the new feature;

FIG. 17 further illustrates a code-generating robot that has just generated itself with a new feature that is now also available for other applications; and

FIG. 18 is a flowchart of a method for application development and maintenance according to an embodiment of the invention.

Detailed Description

The present invention solves the foregoing problems and disadvantages by providing a self-generating automatic code generator that maintains continuity with the enterprise applications it creates throughout development and maintenance of the applications. If a new feature is required by one application that would also by needed by the other applications within the organization, a programmer may simply add this new feature to the self-generating automatic code generator, which is itself an enterprise application, after which the self-generating automatic code generator can, within seconds, regenerate the other applications to include this new feature, including itself.

Reference is now made to the figures in which like reference numerals refer to like elements. For clarity, the first digit of a reference numeral indicates the figure number in which the corresponding element is first used.

In the following description, numerous specific details of programming, software modules, user selections, network transactions, database queries, database structures, etc., are provided for a thorough understanding of the embodiments of the invention. However, those skilled in the art will recognize that the invention can be practiced without one or

more of the specific details, or with other methods, components, materials, etc.

In some cases, well-known structures, materials, or operations are not shown or described in detail in order to avoid obscuring aspects of the invention. Furthermore, the described features, structures, or characteristics may be combined in any suitable manner in one or more embodiments.

FIG. 1 is a block diagram of a self-generating automatic code generator 100 (also referred to herein as a "code-generating robot" or simply "robot") according to an embodiment of the invention. A typical enterprise application essentially includes input, data stores and output. Input data is usually captured by a screen (user interface) or by integration with other applications. In order to make sure that the input data are stored properly, a set of rules must be defined and applied to the input data. The output data is usually a report, a document, a file, or integration with other applications. There is usually a criteria screen for the selection of the output data. A set of rules is also defined and applied to the output data to make them useful.

As shown in FIG. 1, the code-generating robot 100 is, itself, an enterprise application. In one embodiment, the code-generating robot 100 includes a front-end user interface 102 for software developers to capture dynamic knowledge and dynamic rules as input required for automatic code generation. The code-generating robot 100 also includes a dynamic rule layer 104 to apply dynamic rules to captured dynamic knowledge and rules.

As illustrated, the front-end user interface 102, as well as various other screens depicted in succeeding figures, may be displayed within the context of a standard web browser, such as Microsoft Internet Explorer™. However, those of skill in the art will recognize that the present invention is not limited to web-based implementations.

In the depicted embodiment, the code-generating robot 100 includes one or more data stores 106, such as a SQL (structured query language) server, for storing and organizing the dynamic knowledge and dynamic rules captured by the front-end user interface 102. In one embodiment, the SQL server may be embodied as a standard Microsoft SQL Server or other SQL server known to those of skill in the art.

As further illustrated in FIG. 1, the code-generating robot 100 also includes a code generation rule layer 108 to apply static knowledge and static rules to stored dynamic knowledge. Finally, the robot 100 includes, in one configuration, a back-end user interface

110 to generate computer code based on static knowledge and static rules applied to stored dynamic knowledge.

Each enterprise application has its purpose. The purpose of the code-generating robot 100 is to generate an enterprise application such as itself. Therefore, as shown in FIG. 2, the input data is the dynamic knowledge 200 required by the target enterprise application. As an example, for a code-generating robot 100 to generate a screen 202 for a target enterprise application, including itself, the dynamic knowledge 200 captured may include:

- Screen name;
- Table names to store the input data;
- Column names and their structures to store the input data;
- The absolute or relative positions these columns should appear on the screen;
- The manner these columns should appear on the screen;
- etc.

Referring to FIG. 3, depending on the dynamic knowledge 200 captured, certain dynamic rules 300 are provided to ensure that the dynamic knowledge 200 captured is useful data and structured in a desirable manner. Various types of dynamic rules 300 may be captured. For instance, the dynamic rules 300 may include:

- Validation rules governing input data integrity, *e.g.*, constraint rules. An example of a validation rule may include "execution order entered must be positive".
- Calculation rules performing mathematical computation to input data, *e.g.*, computation rules. An example of a calculation rule may include "total invoice amount = sales amount + sales tax".
- Inferring rules modify or add other data based on input data, *e.g.*, inference rules. An example of an inferring rule may include "if new column entered then default new column to read only".
- Action rules triggers other activities based on input data, *e.g.*, action-enabling rules. An example of an action rule may be to delete all the storage of help message, heading, label, etc. associated with a column if that column is being erased from the screen to be generated.

FIG. 3. provides an example screen 302 showing different dynamic rules 300. However, those of skill in the art will recognize that a wide variety of other rules 300 may be provided within the scope of the invention.

Because the code-generating robot 100 is to generate an enterprise application such as itself, several conditions should be satisfied, although not all of the following conditions may be satisfied in every embodiment. First, the application and the code-generating robot 100 employ substantially identical (*i.e.*, similar, compatible, interoperable) user interfaces to capture knowledge and rules. For instance, Screen Columns Maintenance screen Fig. 200 that capture dynamic knowledge for screen columns has essentially the same user interfaces as the Server Rules Maintenance screen Fig. 300 that capture dynamic knowledge for screen business rules.

Second, the knowledge and rules captured by both the application and the code-generating robot are of substantially identical types. As an example, screen column name is stored as Unicode characters.

Third, the application and the code-generating robot 100 employ substantially identical platforms, languages, and architectures. For example, in the depicted embodiment, the application and the code-generating robot 100 may each use Windows Server 2003 as the platform, C# as the computer language, and Microsoft Dot Net Framework as the architecture.

Finally, the application and the code generators employ substantially identical data stores and data relationships. For example, SQL relational databases could be used within an embodiment of the invention.

Referring to FIG. 4, the captured dynamic knowledge 200 and rules 300 are then stored as relational data 400 in an SQL Server (not shown). As illustrated, the relational data 400 may include various tables 402, such as a screen table 402a, a screen object table 402b, a screen type table 402c, a server rule table 402d, and a server rule type table 402e.

In one embodiment, the screen table 402a is used for capturing characteristic of a screen. It may include various columns, such as Screen Type, Table Name, Program Name, etc.

The screen object table 402b is used for capturing columns characteristic of a screen. Columns for the screen object table 402b may include Column Name, Column Type, Column Size, etc..

The screen type table 402c may be used to capture name and description of each screen type. Columns for the screen type table 402c may include Screen Type Name, Screen Type Description, etc.

In the illustrated embodiment, the server rule table 402d may be used to capture server side business rules. Columns within the server rule table 402d may include Rule Name, Rule Type, Execution Order, etc..

Finally, the server rule type table 402 may be used to capture name and description each server rule type. Columns within the server rule type table 402 may include Server Rule Type Name, Server Rule type Description.

In the depicted embodiment of FIG. 4, the relational data 400 is being displayed by a SQL Server Manager, utility 404, which is typically included with a standard SQL server. Such utilities 404 may be used in the process enterprise software development, but are not necessary for the operation of the code-generating robot 100.

Referring to FIG. 5, various static universal knowledge and rules 500 may be applicable to all (or a majority of) applications. The knowledge is generally framework- or platform-related, such as network connectivity, database connectivity, security, language, architecture, etc. For example, in the illustrated embodiment, the static universal knowledge and rules 500 relate to preparing codes based on various data types specific to the database used.

The static universal knowledge and rules 500 are depicted within a SQL query analyzer 502, which, again, is a utility provided with a standard SQL server that may assist with enterprise software development, but is not required for the operation of the code-generating robot 100.

The static universal knowledge and rules 500 usually comprise the code to be generated depending on the dynamic knowledge 200 being captured. As explained with reference to FIG. 1, the code generation rule layer 108 applies the static knowledge and rules 500 to the captured dynamic knowledge 200 and rules 300 that are stored within the data store 106.

As shown in FIG. 6, when a user is ready to generate the computer code required for the target application, including the code-generating robot 100 itself, the back-end user interface 110 for code generation is called upon. The back-end user interface 110 displays a list of screens from which the user may select the screen (or screens) to be generated.

In one embodiment, a selection mechanism 602 is provided to allow the user to decide whether the code is to be generated in the target language and platform on a test basis for further checking or directly deployed to the development environment. When a user selects the "test" option, the system proceeds to generate codes to a predefined directory for reviewing and testing only. If, however, the user selects the "deployment" option, the system proceeds to generate codes to a predefined development directory for compilation.

In one embodiment, the back-end user interface 110 includes an "all screens" check box 604 that allows the user to automatically select all of the listed screens for generation. The generation process starts in response to the user selecting a "create" button 606 or the like.

The computer code generated is not limited to a particular user interface, network interface, security, connectivity, integration, database schema, etc. The computer code generated is also not limited to any computer language, architecture, or platform.

FIG. 7 is flowchart of a process for creating a code-generating robot 100 according to an embodiment of the invention. Those of skill in the art will recognize that the code-generating robot 100 cannot generate itself when it does not initially exist. Accordingly, the process begins with seeding. The first utility that needs to be generated is the screen for capturing dynamic knowledge, as depicted in FIG. 2. Therefore, to seed the code-generating robot 100, the dynamic knowledge that should have been captured by this interface (that does not yet exist) is manually entered 702 and stored in a dynamic knowledge data store 704.

The dynamic knowledge is manually entered with extreme care so that the dynamic rules required for this screen can be deferred. Thereafter, the static knowledge and static rules are manually programmed 706 to apply to the manually entered dynamic knowledge. Using these knowledge and rules, the code generator generates 708 (compiles the knowledge and rules into) the very first screen 710.

This first screen 710 does not yet have any dynamic rules attached, and many other features may still be needed for this screen. Therefore, as shown in FIG. 8, a second screen 802—the user interface needed to capture dynamic rules—is created by using the first screen 710 to capture the dynamic knowledge required for the second screen. This is done carefully, since no dynamic rules are available yet for the first screen 710.

As before, the dynamic knowledge is stored within the dynamic knowledge store 704. Thereafter, the static knowledge and static rules are manually programmed 806 to apply to the manually entered dynamic knowledge. Using these knowledge and rules, the code generator then generates 808 target computer codes for the second screen 802. When the second screen 802 (an example of which is shown in FIG. 3) is available, it is used to capture the dynamic rules required for the first screen 710, and thus, the first screen 710 is enhanced.

By the iterative process of developing more and more utilities as in FIG. 8, more and more new features can be added to the code-generating robot 100. Because the same features are available for the target applications the robot 100 generates, the reverse is also true when a new feature is required by a target application. The new feature is added to this self-generating automatic code generator 100 first. The code for the target application is then generated with the new feature in it.

An example of the foregoing process is now described with reference to FIGs. 9-17. Suppose that a user wishes to add a new feature, such as a filter, to a screen. The purpose of this screen filter is to allow a user to choose from a list of radio-buttons to selectively display certain items from a list of items, *e.g.,* active vs. inactive employees in a payroll system. As illustrated in FIG. 9, selecting the first radio button 902 displays a list of all the active employees, while, as shown in FIG. 10, selecting the second radio-button 1002 displays only the inactive employees.

Referring to FIG. 11, to enable this feature for one or more target applications, such as the one depicted in FIGs. 9 and 10, as well as for the code-generating robot 100 itself, a data store table 1102 called "ScreenFilter" is first created. The table 1102 may include various columns 1104, such as screen filter ID 1104a, screen ID 1104b, screen filter name 1104c, screen filter description 1004d, filter clause 1104e, and filter order 1104f.

The screen filter ID 1104a may be used for identifying this particular screen filter in the system. The screen ID 1104b relates to the particular screen this screen filter applies to. The screen filter name 1104c is the unique name identifying this screen filter. The screen filter description 1104d may contain description other than just a name to easily identify this screen filter upon sorting or searching. The filter clause 1104e relates to the condition this screen filter becomes valid. Finally, the filter order 1104f defines the order

this screen filter appears with respect to other filters for the same screen. Of course, the table 1102 may include additional or different information within the scope of the invention.

Thereafter, as depicted in FIG. 12, the dynamic knowledge needed for the screen "ScreenFilter" is captured using the first screen created by the code-generating robot 100. The dynamic knowledge may be represented within one or more columns 1202. For example, the columns 1202 may include screen column ID 1202a, master table 1202b, column ID 1202c, column name 1202d, default value 1202e, system value 1202f, launch screen 1202g, column size 1202h, column height 1202i, and display mode 1202j, drop down key column 1202k.

The screen column ID 1202c relates to the column of the data store for this column. The column name 1202d is the unique name referring to the column of the data store for this column. The default value 1202e capture the initial displayed value for this column. The system value 1202f capture the server assigned value for this column, etc.. Different or additional columns 1202 may be provided within the scope of the invention.

As shown in FIG. 13, the code for the "ScreenFilter" screen may be generated using the depicted code-generation screen 1300. As previously noted, the code-generation screen 1300 may include a selection mechanism 1302 for allowing the user to determine whether the code is to be generated in the target language and platform on a test basis for further checking or directly deployed to the development environment.

Referring to FIG. 14, note that the new feature "Screen Filter" is not available for the newly generated screen which is used to capture dynamic knowledge for screen filters. As depicted in FIG. 14, this screen is now available to capture dynamic knowledge for all screen filters for selected screens. The "salary" screen is used as an example to demonstrate the knowledge captured. Note that there is no screen filter yet for this screen.

FIG. 15 illustrates the addition of the static knowledge and rules 1500 needed for the generation of codes for this new feature. In the illustrated embodiment, the static knowledge and rules is represented as program code in the C# language. However, any language could be used within the scope of the invention.

Referring to FIG. 16, the next step is to generate and test the codes using salary info screen (see FIG. 9 & 10) as an example. When satisfied, a Screen Filter Maintenance screen 1600 is used to capture dynamic knowledge for itself. The dynamic knowledge

captured for itself includes the screen that refers to itself, the condition that this screen filter becomes valid, and the order this screen filter appears with respect to other screen filters for the same screen.

As shown in FIG. 17, this new filter feature is now apparent as more dynamic knowledge is captured for this Screen Filter Maintenance screen 1700 itself, which is being used to generate other applications. This new feature is fully utilized now in the code-generating robot generator 100. Notice not only the screen (FIG. 17) for the self-generated code generator 100 has the same features as the screen (FIG. 9 & 10) for the application it generates, they share the same look and feel, architecture, language, platform, etc.

FIG. 18 is a flowchart for a method 1800 for adding a new feature to an application generated by an automatic code generator. Initially, a set of knowledge and rules that define an application is captured 1802. Thereafter, code for the application is generated 1804 by providing a first automatic code generator, such as the code-generating robot 100 of FIG. 1, with the set of knowledge and rules.

A set of knowledge and rules that define a new feature to be added to the application is then captured 1806. In one embodiment, code for a second automatic code generator (*e.g.*, a new version of the first automatic code generator) that supports the new feature is then generated 1808 by providing the first automatic code generator with the set of knowledge and rules defining the new feature as well as a set of knowledge and rules defining the first automatic code generator.

Finally, code for the application is regenerated 1810 such that the application includes the new feature by providing the second automatic code generator with the set of knowledge and rules for generating the application as well as a set of knowledge and rules for using the new feature.

While specific embodiments and applications of the present invention have been illustrated and described, it is to be understood that the invention is not limited to the precise configuration and components disclosed herein. Various modifications, changes, and variations apparent to those of skill in the art may be made in the arrangement, operation, and details of the methods and systems of the present invention disclosed herein without departing from the spirit and scope of the present invention.

CLAIMS

1. A computer-implemented method for adding a new feature to an application generated by an automatic code generator comprising:

capturing a set of knowledge and rules that define the application;

generating codes for the application by providing a first automatic code generator with the set of knowledge and rules;

capturing a set of knowledge and rules that define a new feature to be added to the application;

generating codes for a second automatic code generator that supports the new feature by providing the first automatic code generator with the set of knowledge and rules defining the new feature as well as a set of knowledge and rules defining the first automatic code generator; and

regenerating the code for the application such that the application includes the new feature by providing the second automatic code generator with the set of knowledge and rules for generating the application as well as a set of knowledge and rules for using the new feature.

2. The method of claim 1, wherein the application and the code generators employ substantially identical user interfaces to capture knowledge and rules.

3. The method of claim 2, wherein the knowledge and rules captured by both the application and the code generator are of substantially identical types.

4. The method of claim 1, wherein the application and the code generators employ substantially identical platforms, languages, and architectures.

5. The method of claim 1, wherein the application and the code generators employ substantially identical data stores and data relationships.

6. The method of claim 1, wherein the application and the code generators employ substantially identical user interfaces, platforms, languages, and architectures, data stores, and data relationships.

7. The method of claim 1, wherein the first code generator is a subsequent iteration of an earlier code generator automatically generated by the earlier code generator.

8. The method of claim 1, further comprising:

regenerating codes for a second application such that the second application includes the new feature by providing the second automatic code generator with a set of knowledge and rules for generating the second application as well as a set of knowledge and rules for using the new feature.

9. The method of claim 1, wherein the rules comprise at least one of:

validation rules governing input data;

calculation rules for performing mathematical calculations on input data;

inference rules for modifying or adding other data based on input data; and

action rules for triggering actions based on input data.

10. The method of claim 1, further comprising storing captured rules and knowledge within a relational database.

11. A self-generating automatic code generation system comprising:

a user interface for capturing a set of knowledge and rules that define an application;

a first automatic code generator for generating codes for the application using the set of knowledge and rules;

a user-interface for capturing a set of knowledge and rules that define a new feature to be added to the application; and

a second automatic code generator that was automatically generated by the first automatic code generator using the set of knowledge and rules defining the new feature as well as a set of knowledge and rules defining the first automatic code generator,

wherein the second automatic code generator is to regenerate the code for the application such that the application includes the new feature using the set of knowledge and rules for generating the application as well as a set of knowledge and rules for using the new feature.

12. The system of claim 11, wherein the application and the code generators employ substantially identical user interfaces to capture knowledge and rules.

13. The system of claim 12, wherein the knowledge and rules captured by both the application and the code generator are of substantially identical types.

14. The system of claim 11, wherein the application and the code generators employ substantially identical platforms, languages, and architectures.

15. The system of claim 11, wherein the application and the code generators employ substantially identical data stores and data relationships.

16. The system of claim 11, wherein the application and the code generators employ substantially identical user interfaces, platforms, languages, and architectures, data stores, and data relationships.

17. The system of claim 11, wherein the first code generator is a subsequent iteration of an earlier code generator automatically generated by the earlier code generator.

18. The system of claim 11, wherein the second code generator is to regenerate code for a second application such that the second application includes the new feature using a set of knowledge and rules for generating the second application as well as a set of knowledge and rules for using the new feature.

19. The system of claim 11, wherein the rules comprise at least one of:
validation rules governing input data;
calculation rules for performing mathematical calculations on input data;
inference rules for modifying or adding other data based on input data; and
action rules for triggering actions based on input data.

20. The system of claim 11, further comprising a relational database for storing captured rules and knowledge.

21. A computer program product comprising program code for performing a method for adding a new feature to an application generated by an automatic code generator comprising:

program code for capturing a set of knowledge and rules that define the application;

program code for generating codes for the application by providing a first

automatic code generator with the set of knowledge and rules;

program code for capturing a set of knowledge and rules that define a new feature to be added to the application;

program code for generating codes for a second automatic code generator that supports the new feature by providing the first automatic code generator with the set of knowledge and rules defining the new feature as well as a set of knowledge and rules defining the first automatic code generator; and

program code for regenerating the code for the application such that the application includes the new feature by providing the second automatic code generator with the set of knowledge and rules for generating the application as well as a set of knowledge and rules for using the new feature.

22. The computer program product of claim 21, wherein the application and the code generators employ substantially identical user interfaces to capture knowledge and rules.

23. The computer program product of claim 22, wherein the knowledge and rules captured by both the application and the code generator are of substantially identical types.

24. The computer program product of claim 21, wherein the application and the code generators employ substantially identical platforms, languages, and architectures.

25. The computer program product of claim 21, wherein the application and the code generators employ substantially identical data stores and data relationships.

26. The computer program product of claim 21, wherein the application and the code generators employ substantially identical user interfaces, platforms, languages, and architectures, data stores, and data relationships.

27. The computer program product of claim 21, wherein the first code generator is a subsequent iteration of an earlier code generator automatically generated by the earlier code generator.

28. The computer program product of claim 21, further comprising:

program code for regenerating codes for a second application such that the second application includes the new feature by providing the second automatic code generator

with a set of knowledge and rules for generating the second application as well as a set of knowledge and rules for using the new feature.

29. The computer program product of claim 21, wherein the rules comprise at least one of:

validation rules governing input data;

calculation rules for performing mathematical calculations on input data;

inference rules for modifying or adding other data based on input data; and

action rules for triggering actions based on input data.

30. The computer program product of claim 21, further comprising program code for storing captured rules and knowledge within a relational database.

31. A self-generating automatic code generator comprising:

a front-end user interface to capture dynamic knowledge and rules;

a dynamic rule layer to apply the dynamic rules to the dynamic knowledge;

a data store for storing the dynamic knowledge and rules after application of the dynamic rules by the dynamic rule layer;

a code generation rule layer to apply static knowledge and rules to the stored dynamic knowledge; and

a back-end user interface to generate new computer code for the front-end user interface based on the static knowledge and static rules applied to the stored dynamic knowledge.

32. A method for creating a self-generating automatic code generator comprising:

manually programming dynamic knowledge for a user first interface for capturing dynamic knowledge;

manually programming static knowledge and rules to be applied to the manually-programmed dynamic knowledge;

generating codes for the first user interface using the manually-programmed static knowledge and static rules applied to the manually-programmed dynamic knowledge;

using the first user interface to capture dynamic knowledge for a second user interface for capturing dynamic rules;

manually programming static knowledge and rules to be applied to the manually-

programmed dynamic knowledge for the second user interface;

generating codes for the second user interface using the manually-programmed static knowledge and static rules for the second user interface applied to the manually-programmed dynamic knowledge for the second user interface;

using the first and second user interfaces to capture dynamic knowledge and rules, respectively, for a third user interface for the automatic code generator; and

automatically generating codes for the automatic code generator using the captured dynamic knowledge and rules.

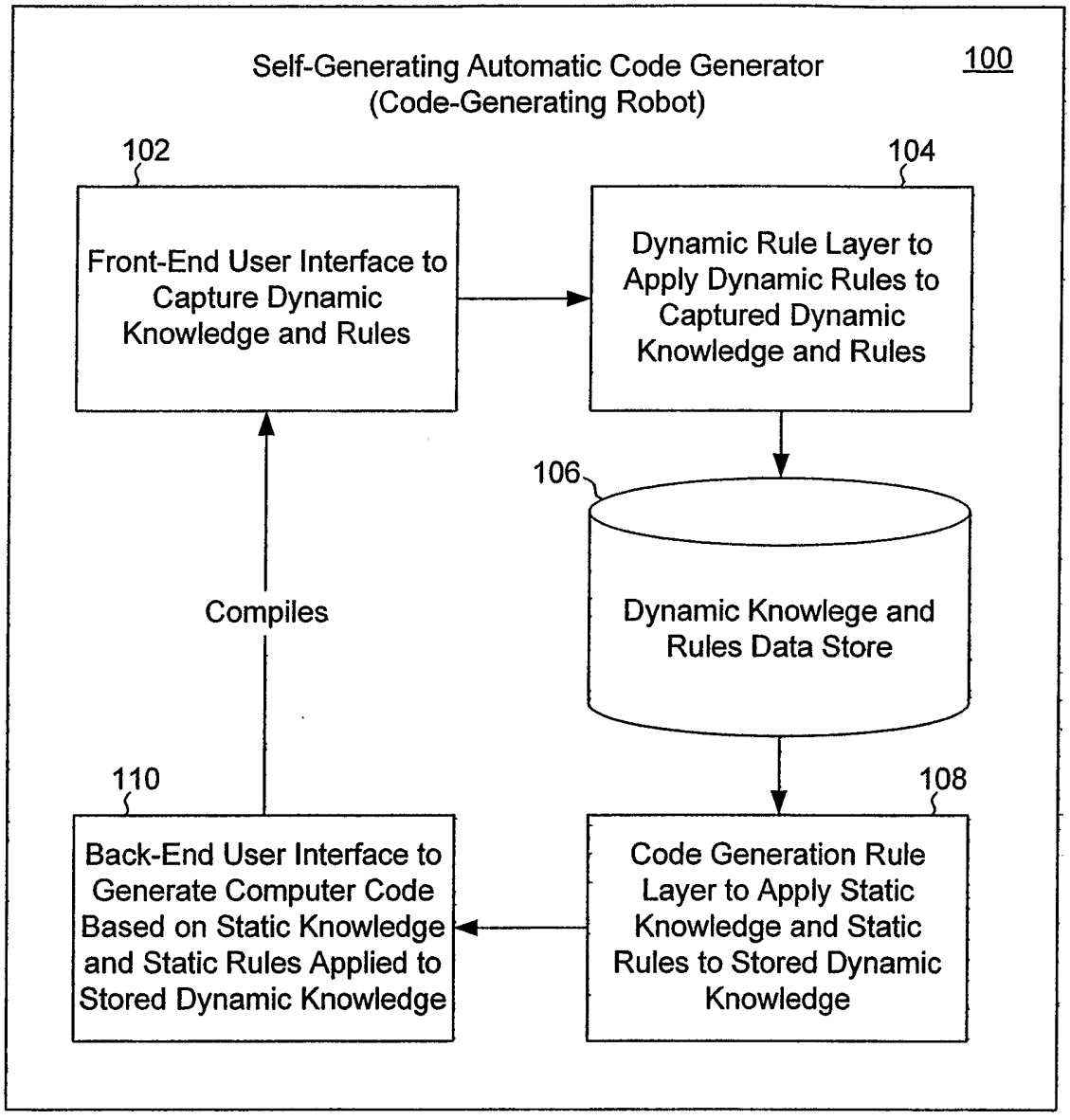

Self-Generating Automatic Code Generator
(Code-Generating Robot)
100
102
Front-End User Interface to Capture Dynamic Knowledge and Rules
104
Dynamic Rule Layer to Apply Dynamic Rules to Captured Dynamic Knowledge and Rules
106
Dynamic Knowlege and Rules Data Store
Compiles
110
Back-End User Interface to Generate Computer Code Based on Static Knowledge and Static Rules Applied to Stored Dynamic Knowledge
108
Code Generation Rule Layer to Apply Static Knowledge and Static Rules to Stored Dynamic Knowledge

FIG. 1

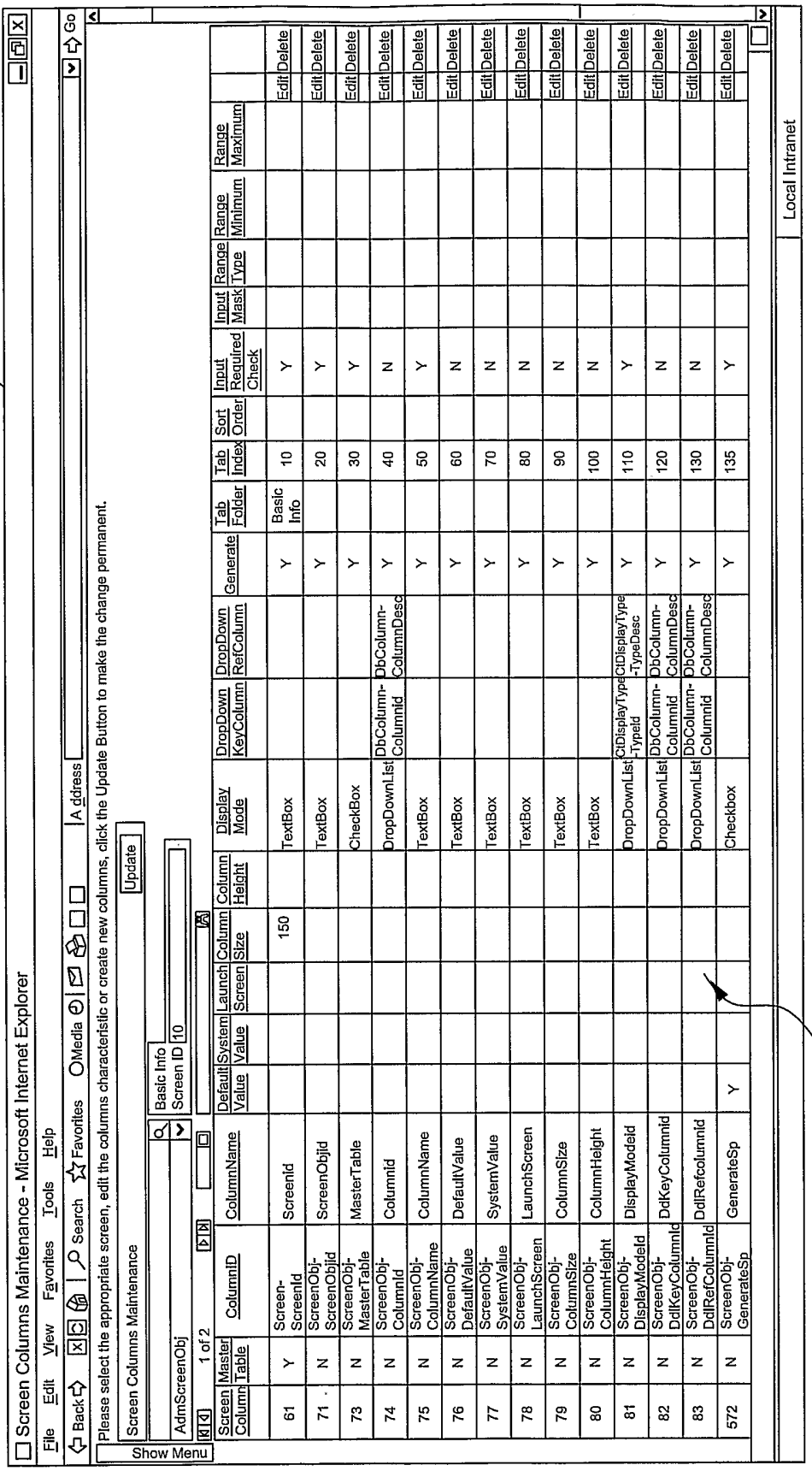

202
Screen Columns Maintenance - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media Address Go
Please select the appropriate screen, edit the columns characteristic or create new columns, click the Update Button to make the change permanent.
Show Menu
Screen Columns Maintenance
Update
AdmScreemObj
Basic Info
Screen ID 10
1 of 2
Screen Column
Master Table
ColumnID
ColumnName
Default Value
System Value
Launch Screen
Column Size
Column Height
Display Mode
DropDown KeyColumn
DropDown RefColumn
Generate
Tab Folder
Tab Index
Sort Order
Input Required Check
Input Mask
Range Type
Range Minimum
Range Maximum
61 Y Screen-ScreenId ScreenId 150 TextBox Y Basic Info 10 Y Edit Delete
71 N ScreenObj-ScreenObjId ScreenObjId TextBox Y 20 Y Edit Delete
73 N ScreenObj-MasterTable MasterTable CheckBox Y 30 Y Edit Delete
74 N ScreenObj-ColumnId ColumnId DropDownList DbColumn-ColumnId DbColumn-ColumnDesc Y 40 N Edit Delete
75 N ScreenObj-ColumnName ColumnName TextBox Y 50 Y Edit Delete
76 N ScreenObj-DefaultValue DefaultValue TextBox Y 60 N Edit Delete
77 N ScreenObj-SystemValue SystemValue TextBox Y 70 N Edit Delete
78 N ScreenObj-LaunchScreen LaunchScreen TextBox Y 80 N Edit Delete
79 N ScreenObj-ColumnSize ColumnSize TextBox Y 90 N Edit Delete
80 N ScreenObj-ColumnHeight ColumnHeight TextBox Y 100 N Edit Delete
81 N ScreenObj-DisplayModeId DisplayModeId DropDownList CtDisplayType-TypeId CtDisplayType-TypeDesc Y 110 Y Edit Delete
82 N ScreenObj-DdlKeyColumnId DdlKeyColumnId DropDownList DbColumn-ColumnId DbColumn-ColumnDesc Y 120 N Edit Delete
83 N ScreenObj-DdlRefcolumnId DdlRefcolumnId DropDownList DbColumn-ColumnId DbColumn-ColumnDesc Y 130 N Edit Delete
572 N ScreenObj-GenerateSp GenerateSp Y Checkbox Y 135 Y Edit Delete
200
Local Intranet

FIG. 2

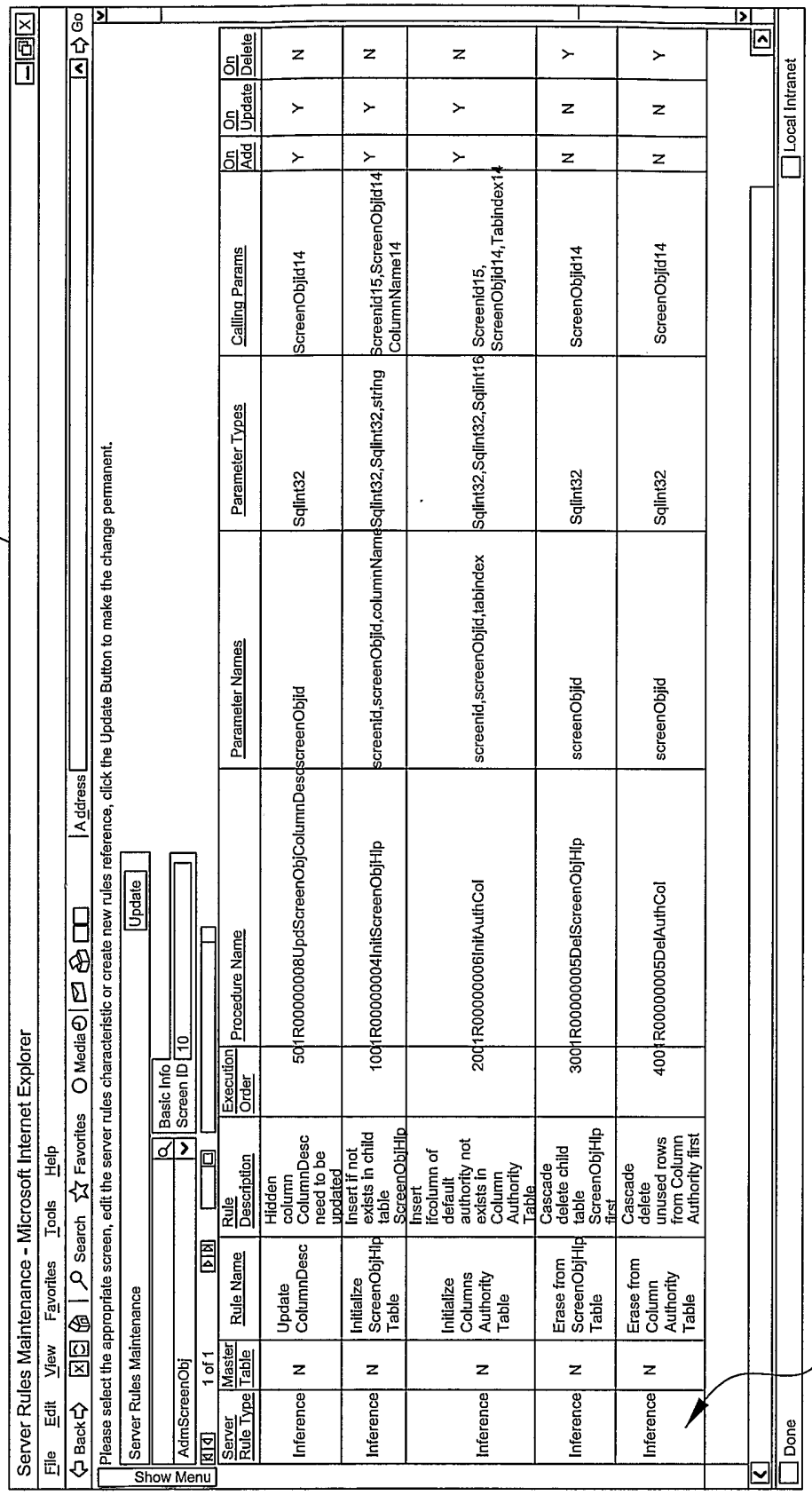

302
Server Rules Maintenance - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media Address Go
Show Menu
Server Rules Maintenance
Update
AdmScreenObj
Basic Info
Screen ID 10
1 of 1
Please select the appropriate screen, edit the server rules characteristic or create new rules reference, click the Update Button to make the change permanent.
Server Rule Type
Master Table
Rule Name
Rule Description
Execution Order
Procedure Name
Parameter Names
Parameter Types
Calling Params
On Add
On Update
On Delete
Inference N Update ColumnDesc Hidden column ColumnDesc need to be updated 50 R00000008UpdScreenObjColumnDesc screenObjId SqlInt32 ScreenObjId14 Y Y N
Inference N Initialize ScreenObjHlp Table Insert if not exists in child table ScreenObjHlp 100 R00000004InitScreenObjHlp screenId,screenObjId,columnName SqlInt32,SqlInt32,string Screenid15,ScreenObjId14 ColumnName14 Y Y N
Inference N Initialize Columns Authority Table Insert ifcolumn of default authority not exists in Column Authority Table 200 R00000006InitAuthCol screenId,screenObjId,tabindex SqlInt32,SqlInt32,SqlInt16 Screenid15, ScreenObjId14,Tabindex14 Y Y N
Inference N Erase from ScreenObjHlp Table Cascade delete child table ScreenObjHlp first 300 R00000005DelScreenObjHlp screenObjId SqlInt32 ScreenObjId14 N N Y
Inference N Erase from Column Authority Table Cascade delete unused rows from Column Authority first 400 R00000005DelAuthCol screenObjId SqlInt32 ScreenObjId14 N N Y
300
Done
Local Intranet

FIG. 3

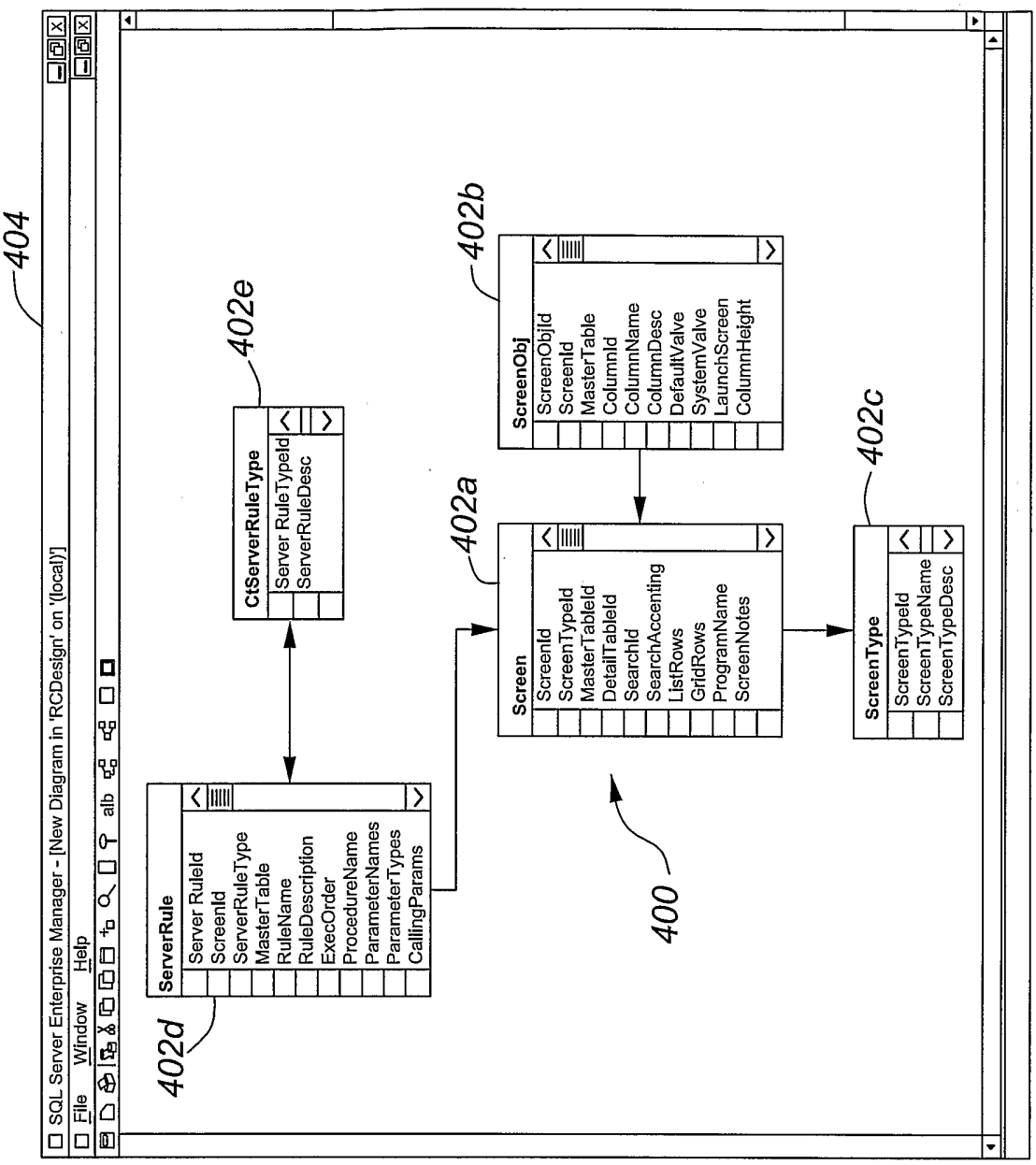

404
SQL Server Enterprise Manager - [New Diagram in 'RCDesign' on '(local)']
File Window Help
402e
CtServerRuleType
Server RuleTypeId
ServerRuleDesc
402b
ScreenObj
ScreenObjId
ScreenId
MasterTable
ColumnId
ColumnName
ColumnDesc
DefaultValve
SystemValve
LaunchScreen
ColumnHeight
402a
Screen
ScreenId
ScreenTypeId
MasterTableId
DetailTableId
SearchId
SearchAccenting
ListRows
GridRows
ProgramName
ScreenNotes
402c
ScreenType
ScreenTypeId
ScreenTypeName
ScreenTypeDesc
ServerRule
Server RuleId
ScreenId
ServerRuleType
MasterTable
RuleName
RuleDescription
ExecOrder
ProcedureName
ParameterNames
ParameterTypes
CallingParams
402d
400

FIG. 4

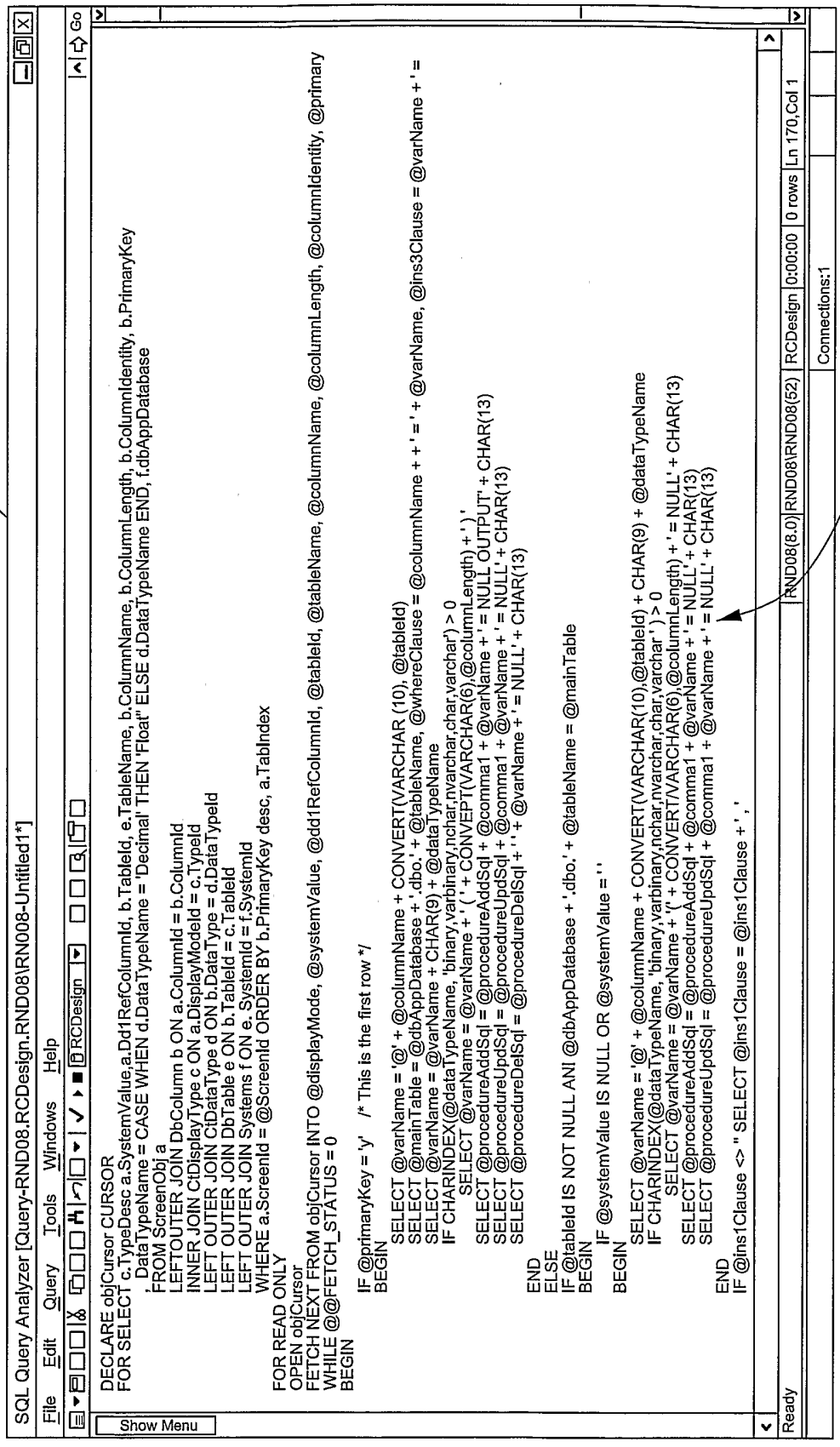

502
SQL Query Analyzer [Query-RND08.RCDesign.RND08\RN008-Untitled1*]
File Edit Query Tools Windows Help
RCDesign
Show Menu
DECLARE objCursor CURSOR
FOR SELECT c.TypeDesc a.SystemValue,a.Dd1RefColumnId, b.TableId, e.TableName, b.ColumnName, b.ColumnLength, b.ColumnIdentity, b.PrimaryKey
,DataTypeName = CASE WHEN d.DataTypeName = 'Decimal' THEN 'Float' ELSE d.DataTypeName END, f.dbAppDatabase
FROM ScreenObj a
LEFTOUTER JOIN DbColumn b ON a.ColumnId = b.ColumnId
INNER JOIN CtDisplayType c ON a.DisplayModeId = c.TypeId
LEFT OUTER JOIN CtDataType d ON b.DataType = d.DataTypeId
LEFT OUTER JOIN DbTable e ON b.TableId = c.TableId
LEFT OUTER JOIN Systems f ON e.SystemId = f.SystemId
WHERE a.ScreenId = @ScreenId ORDER BY b.PrimaryKey desc, a.TabIndex
FOR READ ONLY
OPEN objCursor
FETCH NEXT FROM objCursor INTO @displayMode, @systemValue, @dd1RefColumnId, @tableId, @tableName, @columnName, @columnLength, @columnIdentity, @primary
WHILE @@FETCH_STATUS = 0
BEGIN
IF @primaryKey = 'y' /* This is the first row */
BEGIN
SELECT @varName = '@' + @columnName + CONVERT(VARCHAR (10), @tableId)
SELECT @mainTable = @dbAppDatabase + '.dbo.' + @tableName, @whereClause = @columnName + ' + ' = ' + @varName, @ins3Clause = @varName + ' =
SELECT @varName = @varName + CHAR(9) + @dataTypeName
IF CHARINDEX(@dataTypeName, 'binary,varbinary,nchar,nvarchar,char,varchar') > 0
SELECT @varName = @varName + ' (' + CONVEPT(VARCHAR(6),@columnLength) + ') '
SELECT @procedureAddSql = @procedureAddSql + @comma1 + @varName + ' = NULL OUTPUT' + CHAR(13)
SELECT @procedureUpdSql = @procedureUpdSql + @comma1 + @varName + ' = NULL' + CHAR(13)
SELECT @procedureDelSql = @procedureDelSql + ' ' + @varName + ' = NULL' + CHAR(13)
END
ELSE
IF @tableId IS NOT NULL ANI @dbAppDatabase + '.dbo.' + @tableName = @mainTable
BEGIN
IF @systemValue IS NULL OR @systemValue = ' '
BEGIN
SELECT @varName = '@' + @columnName + CONVERT(VARCHAR(10),@tableId) + CHAR(9) + @dataTypeName
IF CHARINDEX(@dataTypeName, 'binary,varbinary,nchar,nvarchar,char,varchar') > 0
SELECT @varName = @varName + '(' + CONVERT/VARCHAR(6),@columnLength) + ' = NULL' + CHAR(13)
SELECT @procedureAddSql = @procedureAddSql + @comma1 + @varName + ' = NULL' + CHAR(13)
SELECT @procedureUpdSql = @procedureUpdSql + @comma1 + @varName + ' = NULL' + CHAR(13)
END
IF @ins1Clause <> '' SELECT @ins1Clause = @ins1Clause + ' , '
500
Ready
RND08(8.0) RND08\RND08(52) RCDesign 0:00:00 0 rows Ln 170,Col 1
Connections:1

FIG. 5

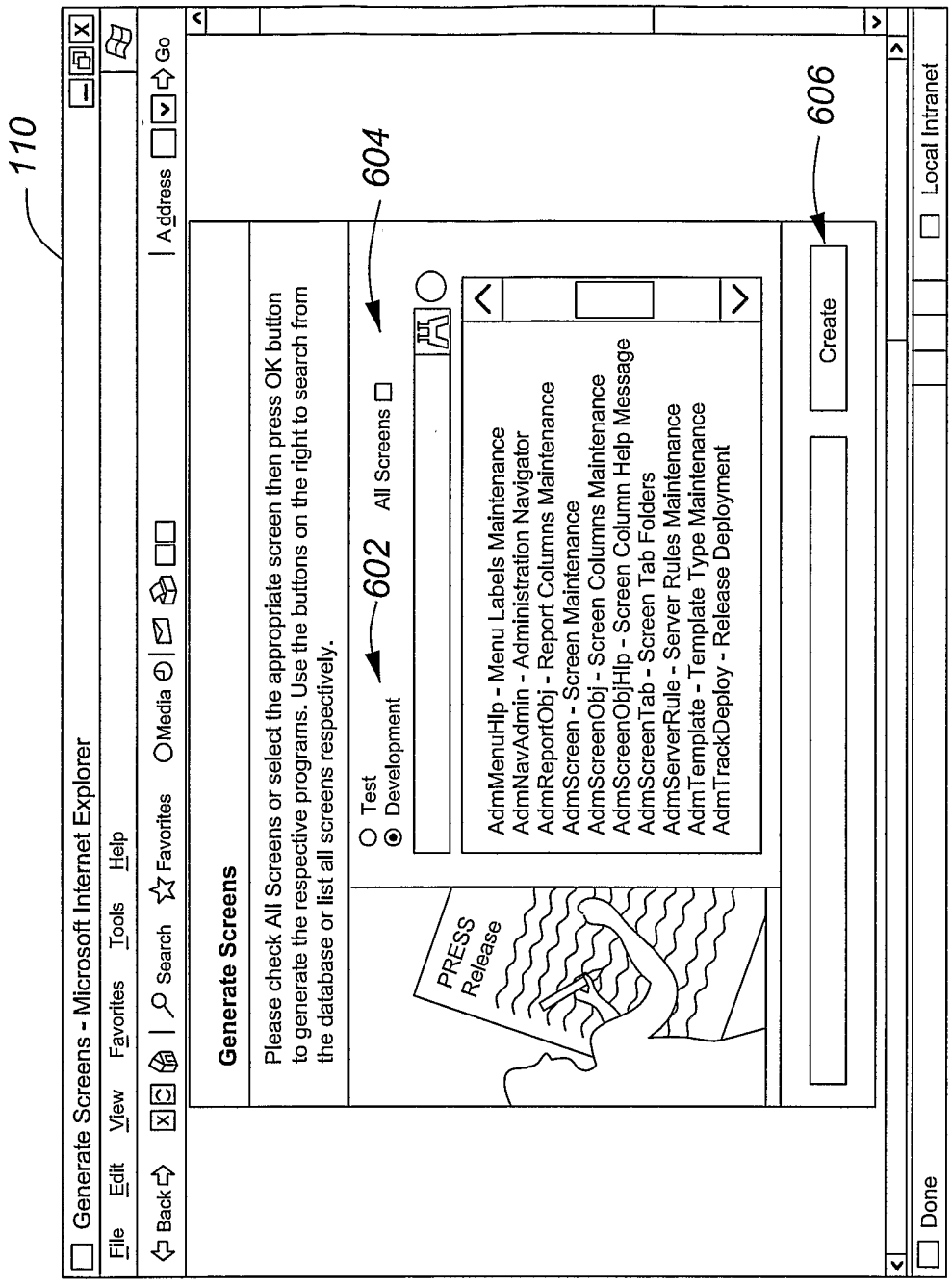

110
Generate Screens - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media Address Go
Generate Screens
Please check All Screens or select the appropriate screen then press OK button to generate the respective programs. Use the buttons on the right to search from the database or list all screens respectively.
602
Test
Development
All Screens
604
AdmMenuHlp - Menu Labels Maintenance
AdmNavAdmin - Administration Navigator
AdmReportObj - Report Columns Maintenance
AdmScreen - Screen Maintenance
AdmScreenObj - Screen Columns Maintenance
AdmScreenObjHlp - Screen Column Help Message
AdmScreenTab - Screen Tab Folders
AdmServerRule - Server Rules Maintenance
AdmTemplate - Template Type Maintenance
AdmTrackDeploy - Release Deployment
PRESS Release
606
Create
Done
Local Intranet

FIG. 6

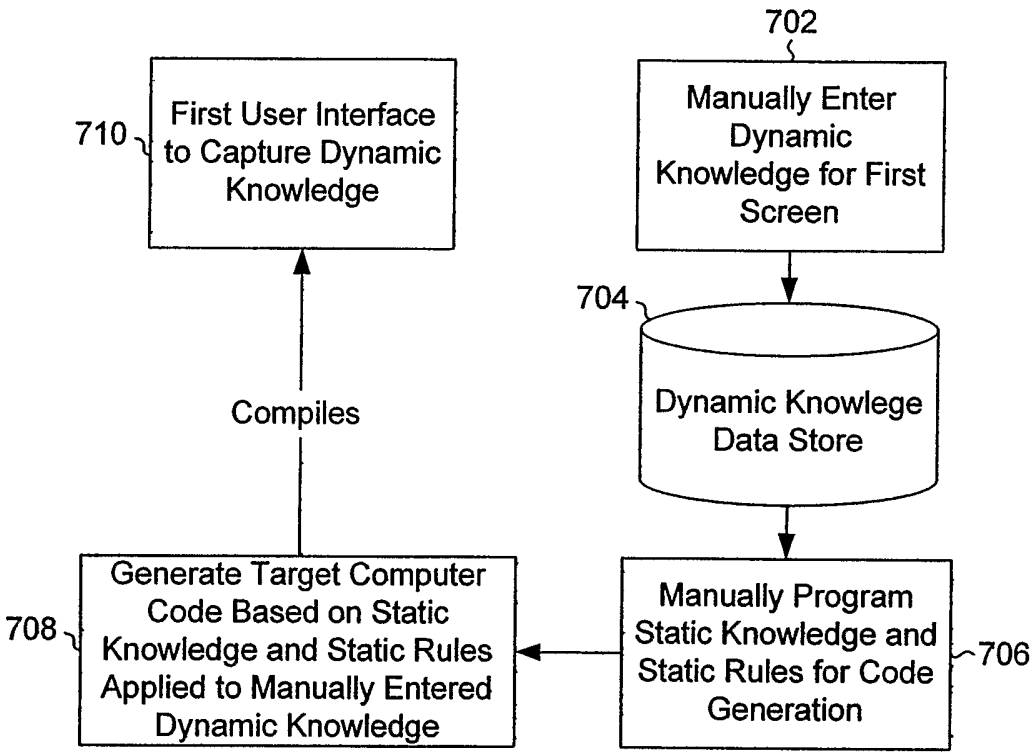

702
Manually Enter Dynamic Knowledge for First Screen
710
First User Interface to Capture Dynamic Knowledge
704
Dynamic Knowlege Data Store
Compiles
Generate Target Computer Code Based on Static Knowledge and Static Rules Applied to Manually Entered Dynamic Knowledge
708
Manually Program Static Knowledge and Static Rules for Code Generation
706

FIG. 7

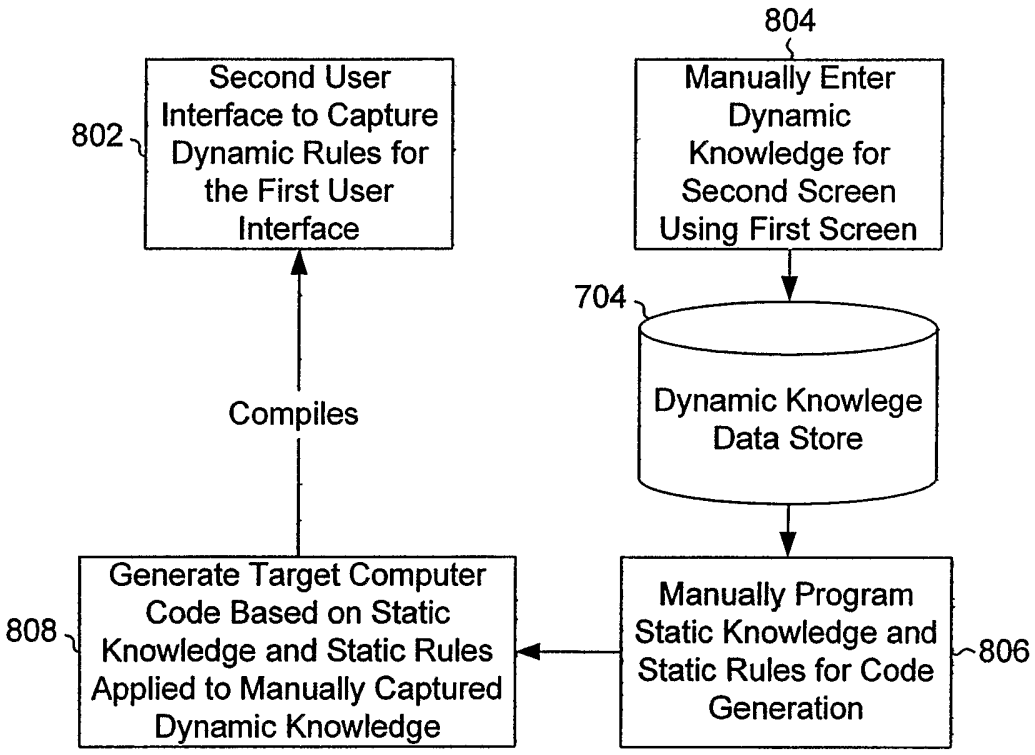

804
Second User Interface to Capture Dynamic Rules for the First User Interface
802
Manually Enter Dynamic Knowledge for Second Screen Using First Screen
704
Dynamic Knowlege Data Store
Compiles
Generate Target Computer Code Based on Static Knowledge and Static Rules Applied to Manually Captured Dynamic Knowledge
808
Manually Program Static Knowledge and Static Rules for Code Generation
806

FIG. 8

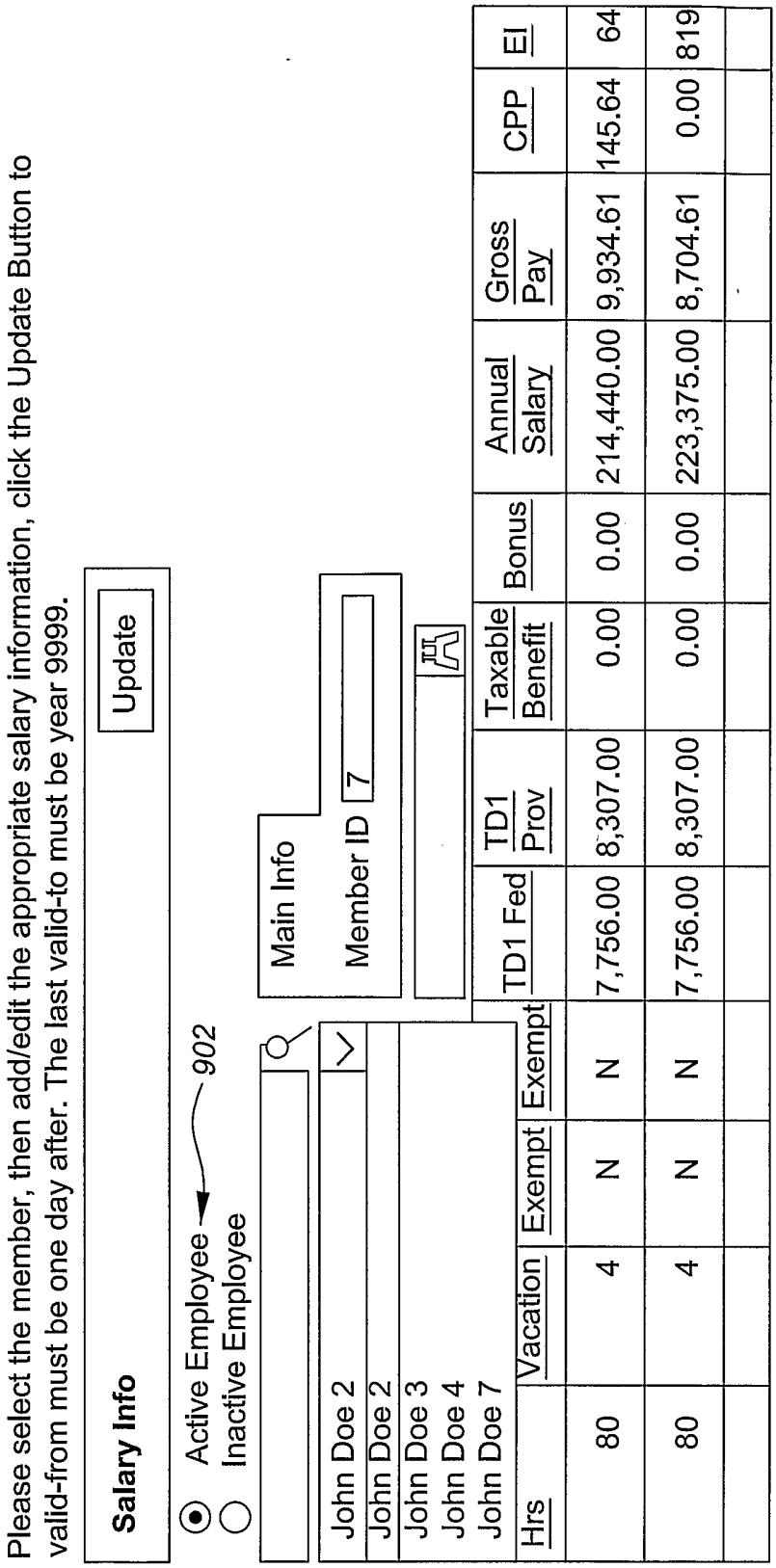
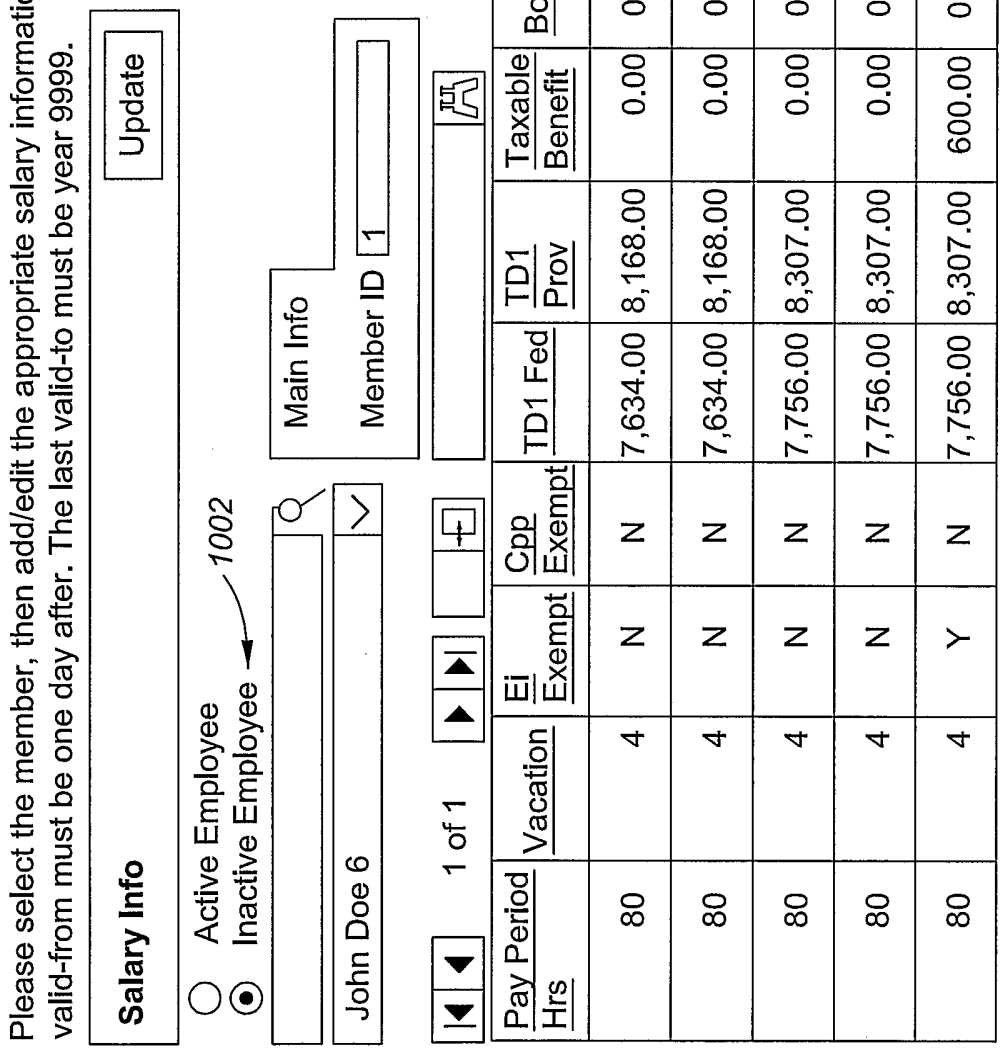

Please select the member, then add/edit the appropriate salary information, click the Update Button to valid-from must be one day after. The last valid-to must be year 9999.
Salary Info
Update
Active Employee
Inactive Employee
902
Main Info
Member ID 7
John Doe 2
John Doe 2
John Doe 3
John Doe 4
John Doe 7
Hrs
Vacation
Exempt
Exempt
TD1 Fed
TD1 Prov
Taxable Benefit
Bonus
Annual Salary
Gross Pay
CPP
EI
80 4 N N 7,756.00 8,307.00 0.00 0.00 214,440.00 9,934.61 145.64 64
80 4 N N 7,756.00 8,307.00 0.00 0.00 223,375.00 8,704.61 0.00 819

FIG. 9



Pay Period Hrs	Vacation	Ei Exempt	Cpp Exempt	TD1 Fed	TD1 Prov	Taxable Benefit	Bonus	Annual Salary	Gross Pay	CPP	EI
80	4	N	N	7,634.00	8,168.00	0.00	0.00	55,754.40	1,476.00	50.07	26
80	4	N	N	7,634.00	8,168.00	0.00	0.00	78,628.00	2,081.54	73.21	37
80	4	N	N	7,756.00	8,307.00	0.00	0.00	78,628.00	2,081.54	77.11	35
80	4	N	N	7,756.00	8,307.00	0.00	0.00	17,220.00	1,838.46	107.57	48
80	4	Y	N	7,756.00	8,307.00	600.00	0.00	39,220.00	9,938.46	0.00	0

FIG. 10

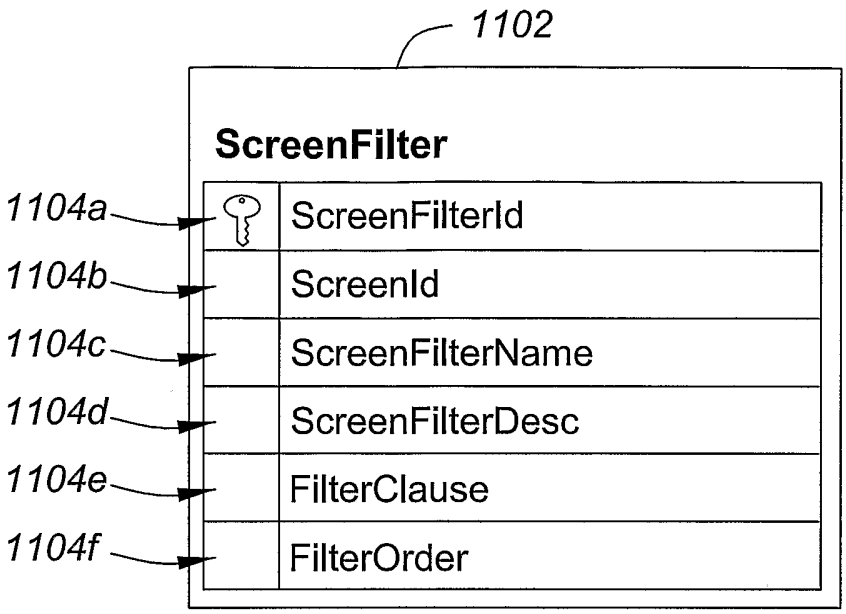
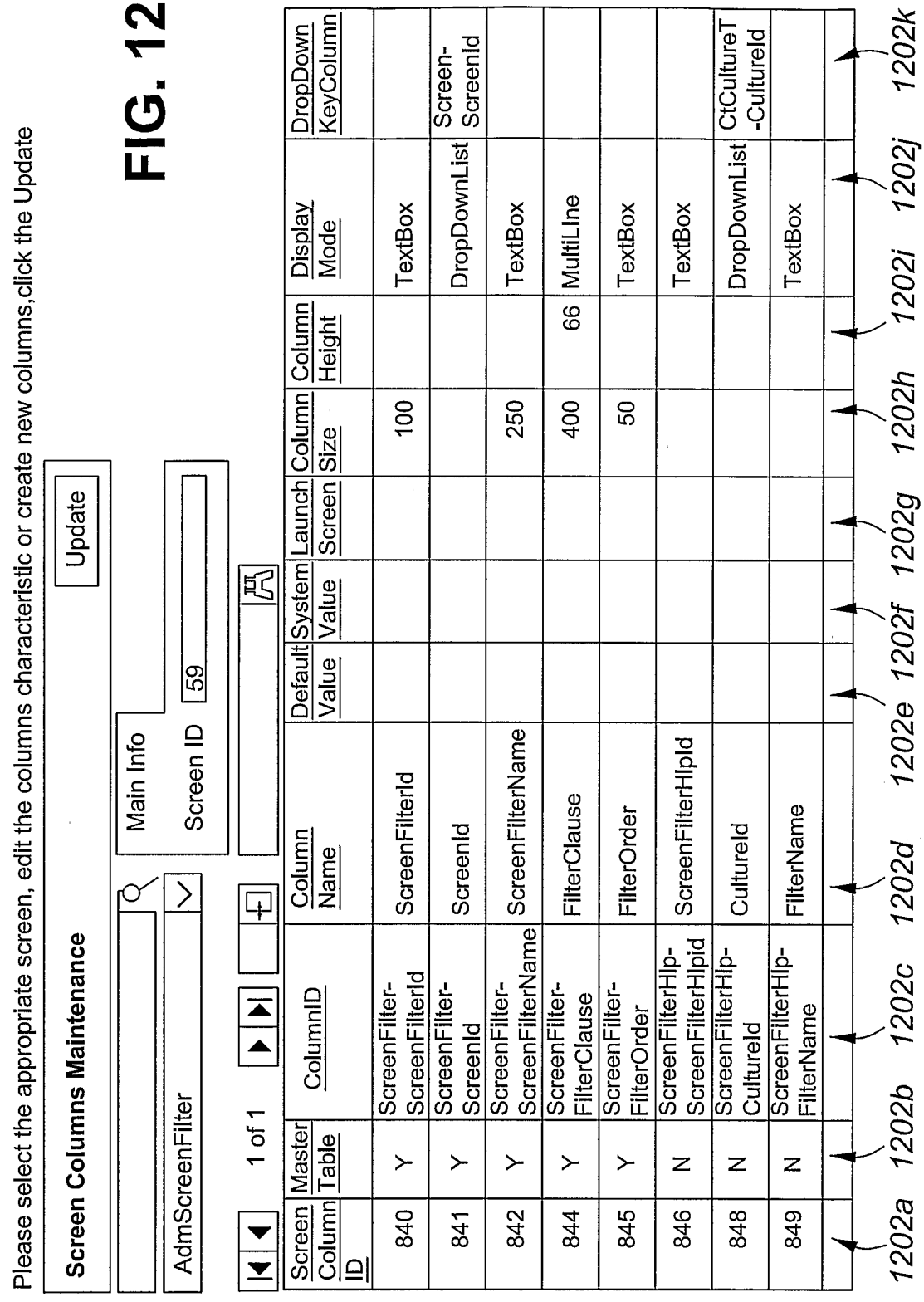

1102
ScreenFilter
1104a
ScreenFilterId
1104b
ScreenId
1104c
ScreenFilterName
1104d
ScreenFilterDesc
1104e
FilterClause
1104f
FilterOrder

FIG. 11

FIG. 12


Please select the appropriate screen, edit the columns characteristic or create new columns,click the Update
Screen Columns Maintenance
Update
Main Info
Screen ID 59
AdmScreenFilter
1 of 1
Screen Column ID
Master Table
ColumnID
Column Name
Default Value
System Value
Launch Screen
Column Size
Column Height
Display Mode
DropDown KeyColumn
840 Y ScreenFilter-ScreenFilterId ScreenFilterId 100 TextBox
841 Y ScreenFilter-ScreenId ScreenId DropDownList Screen-ScreenId
842 Y ScreenFilter-ScreenFilterName ScreenFilterName 250 TextBox
844 Y ScreenFilter-FilterClause FilterClause 400 66 MultilLne
845 Y ScreenFilter-FilterOrder FilterOrder 50 TextBox
846 N ScreenFilterHlp-ScreenFilterHlpId ScreenFilterHlpId TextBox
848 N ScreenFilterHlp-CultureId CultureId DropDownList CtCultureT-CultureId
849 N ScreenFilterHlp-FilterName FilterName TextBox
1202a 1202b 1202c 1202d 1202e 1202f 1202g 1202h 1202i 1202j 1202k

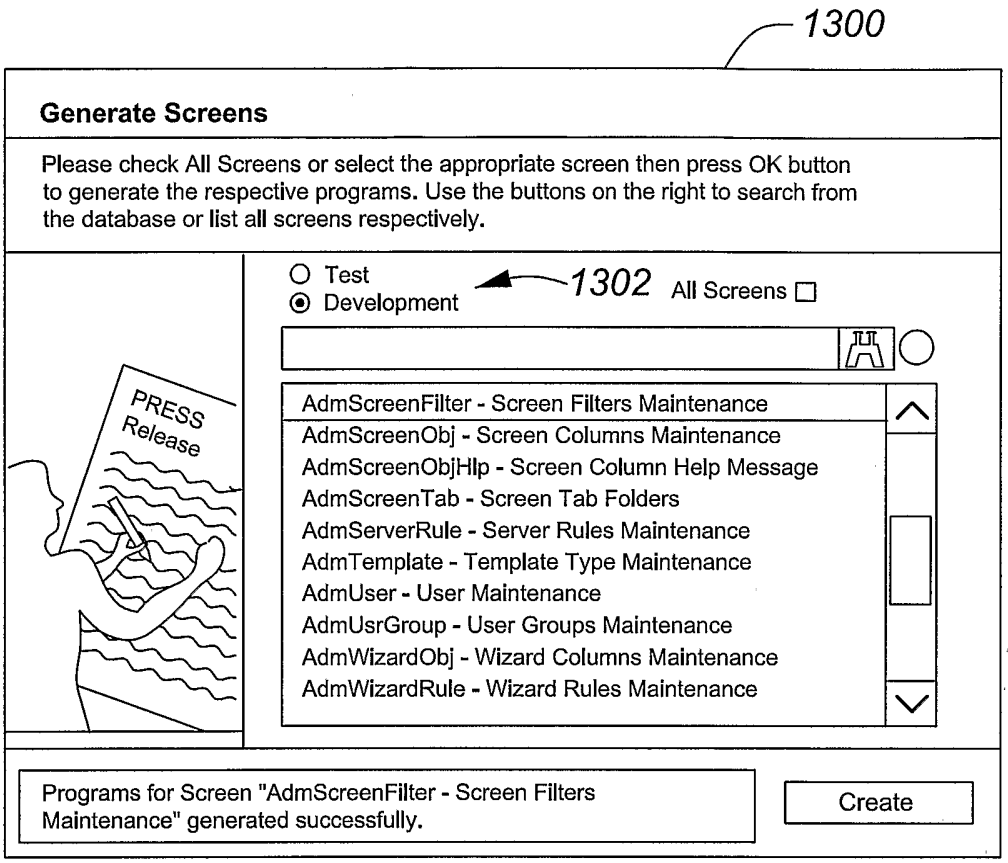

1300
Generate Screens
Please check All Screens or select the appropriate screen then press OK button to generate the respective programs. Use the buttons on the right to search from the database or list all screens respectively.
Test
Development
1302
All Screens
PRESS
Release
AdmScreenFilter - Screen Filters Maintenance
AdmScreenObj - Screen Columns Maintenance
AdmScreenObjHlp - Screen Column Help Message
AdmScreenTab - Screen Tab Folders
AdmServerRule - Server Rules Maintenance
AdmTemplate - Template Type Maintenance
AdmUser - User Maintenance
AdmUsrGroup - User Groups Maintenance
AdmWizardObj - Wizard Columns Maintenance
AdmWizardRule - Wizard Rules Maintenance
Programs for Screen "AdmScreenFilter - Screen Filters Maintenance" generated successfully.
Create

FIG. 13

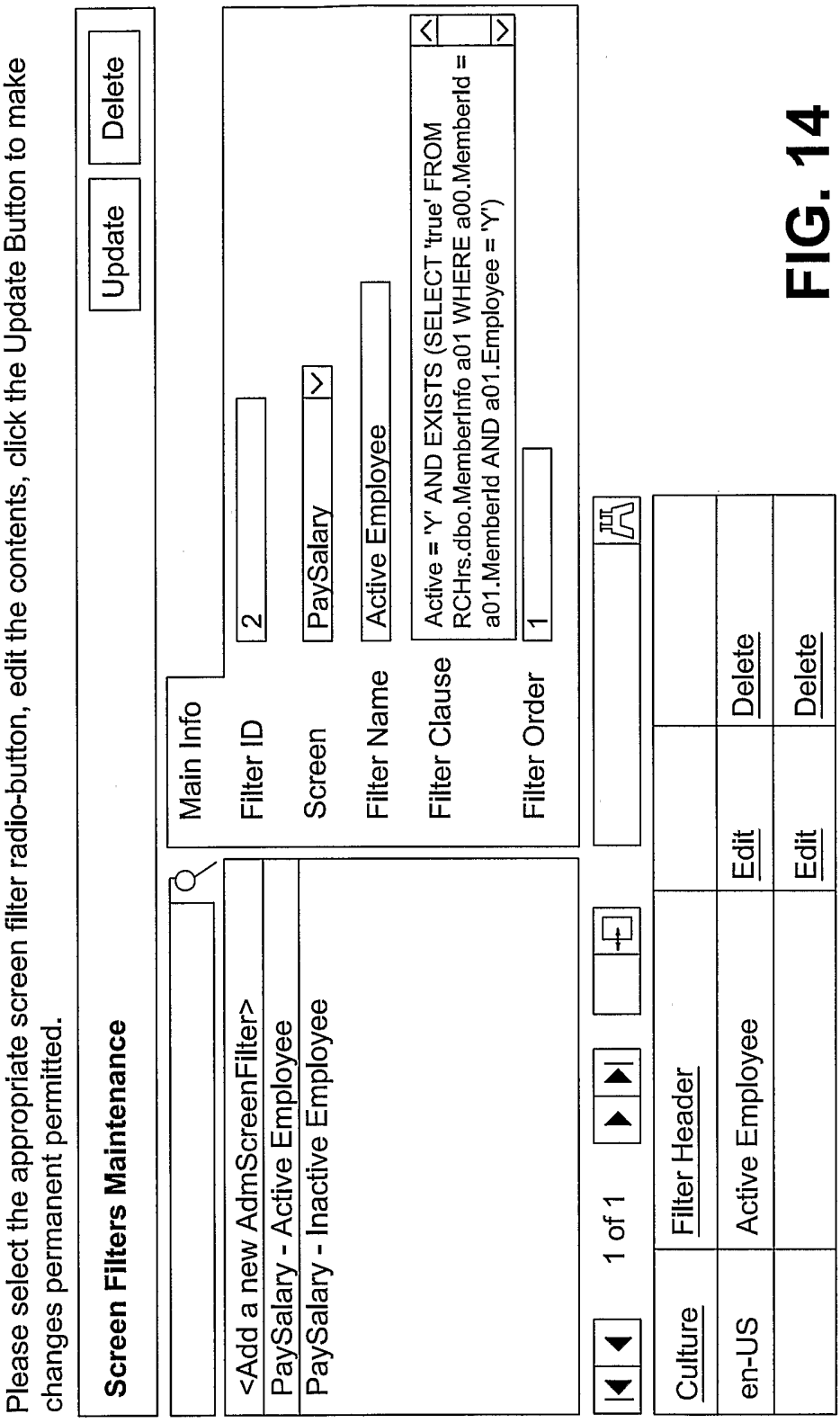

Please select the appropriate screen filter radio-button, edit the contents, click the Update Button to make changes permanent permitted.
Screen Filters Maintenance
Update
Delete
<Add a new AdmScreenFilter>
PaySalary - Active Employee
PaySalary - Inactive Employee
Main Info
Filter ID
2
Screen
PaySalary
Filter Name
Active Employee
Filter Clause
Active = 'Y' AND EXISTS (SELECT 'true' FROM RCHrs.dbo.MemberInfo a01 WHERE a00.MemberId = a01.MemberId AND a01.Employee = 'Y')
Filter Order
1
1 of 1
Culture
Filter Header
en-US
Active Employee
Edit
Delete
Edit
Delete

FIG. 14

1500

```
sb.Append(GetCnDclr(dw["SysProgram"].ToString() ) + ", SqlInt32 screenFilterId)" + (char)13);
sb.Append("        (" + (char) 13) ;
sb.Append("            if ( da == null ) " + (char)13);
sb.Append("            ( ' + (char) 13) ;
sb.Append("                throw new System.ObjectDisposedException( GetType().FullName );" + (char
sb.Append("            )" + (char)13);
sb.Append("            SqlCommand selCmd = new SqlCommand(\"GetLis" + dw["ProgramName"].ToString()
sb.Append("            selCmd.CommandType = CommandType.StoredProcedure;" + (char)13);
parameterNames.Remove(0,parameterNames.Length);
parameterNames.Append(dtLis.Rows[0]["ParameterNames"].ToString() );
param ="";
while  (parameterNames.Length > 0)
(
    param = (new Utils()).PopFirstWord(parameterNames,  (char)44);
    sb.Append("        selCmd.Parameters.Add(\"@" + param + "\", " + param + ");" + (char)13);
)
sb.Append("        selCmd.Parameters.Add(\"@screen Filter Id\", screenFilterId);" + (char)13);
sb.Append("        da.SelectCommand = selCmd;" + (char)13);
sb.Append("        DataTable dt = new DataTable();" + (char)13);
sb.Append("        da.Fill(dt);" + (char)13);
sb.Append("        return dt;" + (char)13);
sb.Append("    )" + (char)13);
```

FIG. 15

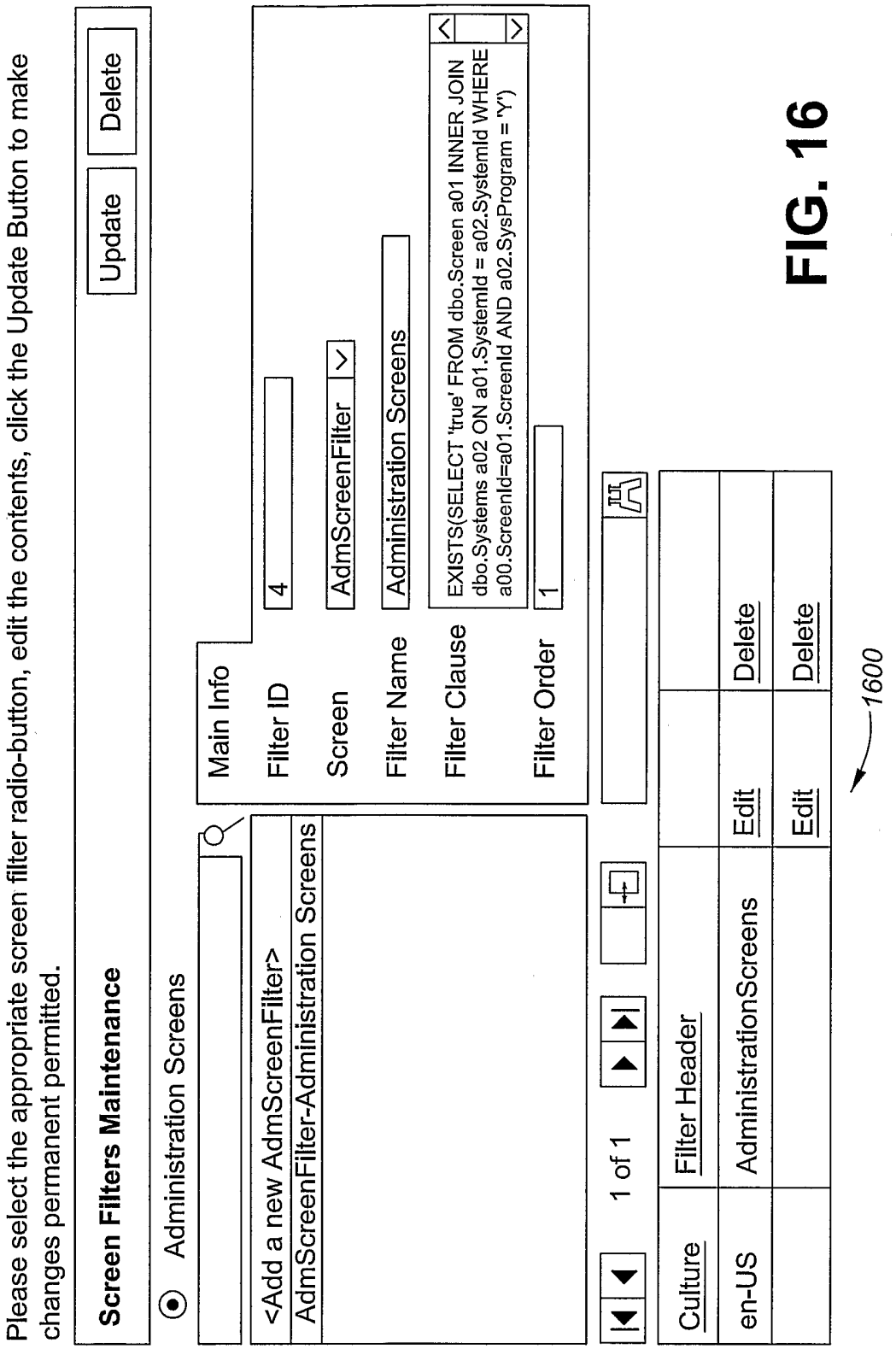

Please select the appropriate screen filter radio-button, edit the contents, click the Update Button to make changes permanent permitted.
Screen Filters Maintenance
Update
Delete
Administration Screens
<Add a new AdmScreenFilter>
AdmScreenFilter-Administration Screens
Main Info
Filter ID
4
Screen
AdmScreenFilter
Filter Name
Administration Screens
Filter Clause
EXISTS(SELECT 'true' FROM dbo.Screen a01 INNER JOIN dbo.Systems a02 ON a01.SystemId = a02.SystemId WHERE a00.ScreenId=a01.ScreenId AND a02.SysProgram = 'Y')
Filter Order
1
1 of 1
Culture
Filter Header
en-US
AdministrationScreens
Edit
Delete
Edit
Delete
1600

FIG. 16

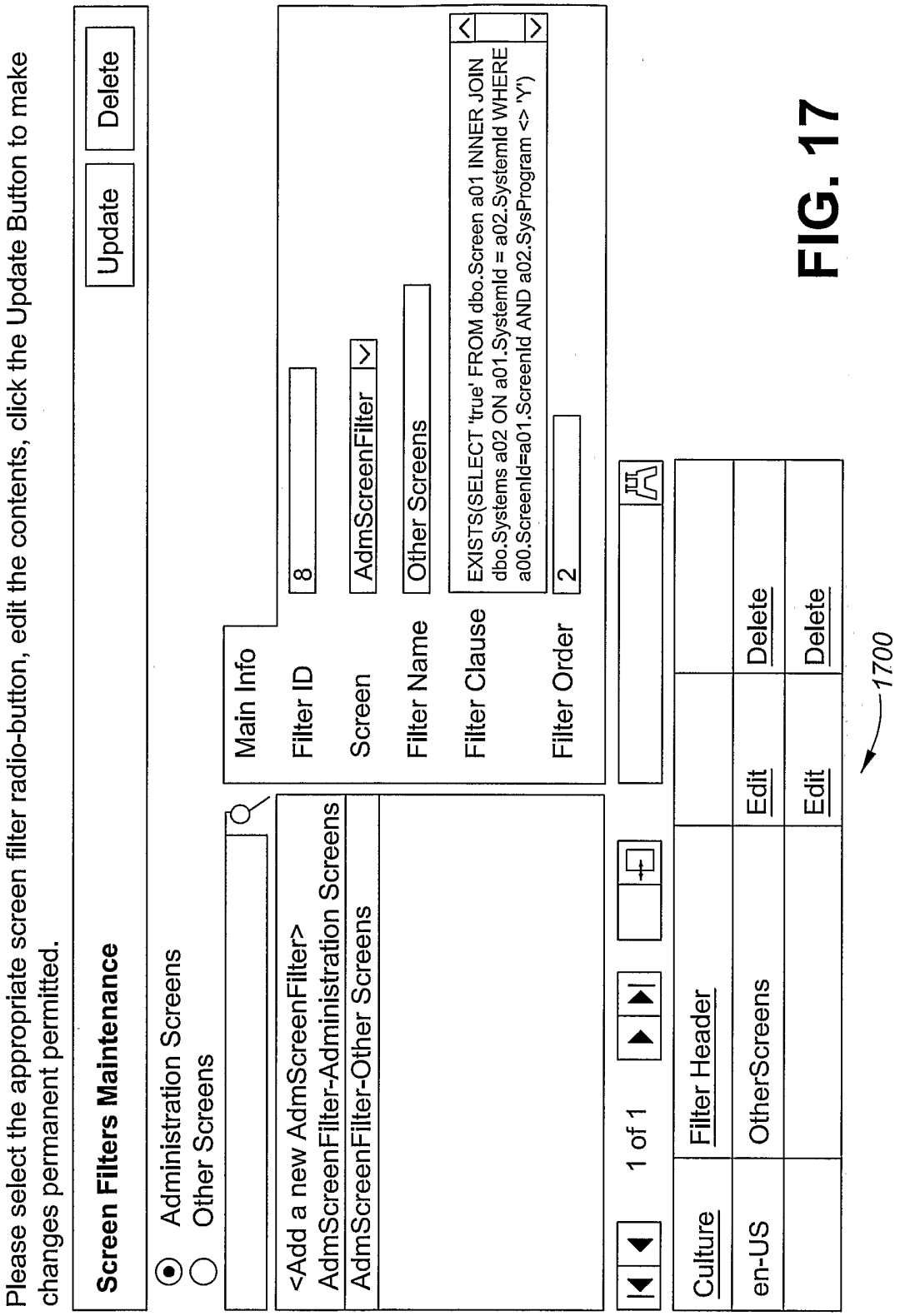

Please select the appropriate screen filter radio-button, edit the contents, click the Update Button to make changes permanent permitted.
Screen Filters Maintenance
Update
Delete
Administration Screens
Other Screens
<Add a new AdmScreenFilter>
AdmScreenFilter-Administration Screens
AdmScreenFilter-Other Screens
Main Info
Filter ID
8
Screen
AdmScreenFilter
Filter Name
Other Screens
Filter Clause
EXISTS(SELECT 'true' FROM dbo.Screen a01 INNER JOIN dbo.Systems a02 ON a01.SystemId = a02.SystemId WHERE a00.ScreenId=a01.ScreenId AND a02.SysProgram <> 'Y')
Filter Order
2
1 of 1
Culture
Filter Header
en-US
OtherScreens
Edit
Delete
Edit
Delete
1700

FIG. 17

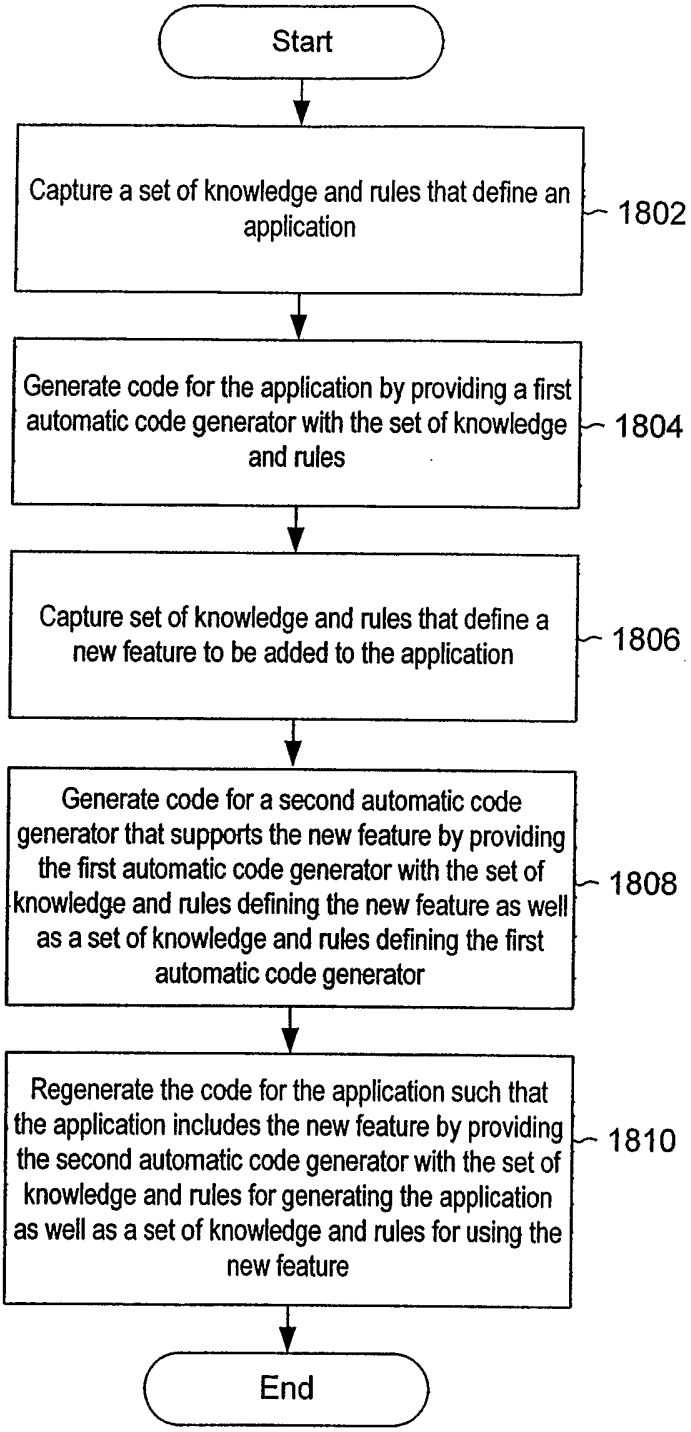

1800
Start
Capture a set of knowledge and rules that define an application
1802
Generate code for the application by providing a first automatic code generator with the set of knowledge and rules
1804
Capture set of knowledge and rules that define a new feature to be added to the application
1806
Generate code for a second automatic code generator that supports the new feature by providing the first automatic code generator with the set of knowledge and rules defining the new feature as well as a set of knowledge and rules defining the first automatic code generator
1808
Regenerate the code for the application such that the application includes the new feature by providing the second automatic code generator with the set of knowledge and rules for generating the application as well as a set of knowledge and rules for using the new feature
1810
End

FIG. 18

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2005/000054

A. CLASSIFICATION OF SUBJECT MATTER

IPC7 G06F 9/44

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC7 G06F

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

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms

Delphion, IEEE Xplore, Canadian Patent Database, & keywords: code generat*, new feature, regenerat*, re-generat*.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant	Relevant to claim No(s).
X	US 20040015809 A1 (Cheng) 22 Jan., 2004 (22-01-2004) abstract, [0005], [0008], [0021], [0022], [0024], [0025], [0030]-[0040], [0042], [0049], [0052] and Figs. 1-4	1-32
A	US 6658579 B1 (Bell et al.) 02 Dec., 2003 (02-12-2003) col.10 lines 19-55, col. 11 lines 30-43, col. 26 lines 16-34	1-32
A	US 6671699 B1 (Black et al.) 30 Dec. 2003 (30-12-2003) col. 7 line 6-col. 8 line 32, col. 9 lines 15-27, col. 23 lines 44-62	1-32

[ ] Further documents are listed in the continuation of Box C. [X] See patent family annex.

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

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

Date of the actual completion of the international search	Date of mailing of the international search report
16 March 2005 (16-03-2005)	31 March 2005 (31-03-2005)

Name and mailing address of the ISA/CA	Authorized officer
Canadian Intellectual Property Office *Place du Portage I, C114 - 1st Floor, Box PCT* *50 Victoria Street* *Gatineau, Quebec K1A 0C9* Facsimile No: 001(819)953-2476	Cristian S. Popa (819) 997-2299

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/CA2005/000054

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
US2004015809	22-01-2004	US2004015809 A1	22-01-2004
		WO02095570 A2	28-11-2002
US6658579	02-12-2003	AU6166301 A	03-12-2001
		AU6167701 A	03-12-2001
		AU6171101 A	03-12-2001
		AU6463801 A	03-12-2001
		EP1299966 A1	09-04-2003
		EP1305686 A2	02-05-2003
		JP2003534748T T	18-11-2003
		JP2004523139T T	29-07-2004
		US6332198 B1	18-12-2001
		US6461513 B1	08-10-2002
		US6639910 B1	28-10-2003
		US6654903 B1	25-11-2003
		US6658579 B1	02-12-2003
		US6658580 B1	02-12-2003
		US6760339 B1	06-07-2004
		US2002001307 A1	03-01-2002
		US2002057018 A1	16-05-2002
		US2002116485 A1	22-08-2002
		US2002118031 A1	29-08-2002
		US2003126195 A1	03-07-2003
		US2004031030 A1	12-02-2004
		WO0189669 A2	29-11-2001
		WO0190843 A2	29-11-2001
		WO0190865 A1	29-11-2001
		WO0191345 A1	29-11-2001
US6671699	30-12-2003	US6671699 B1	30-12-2003