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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 November 2009 (19.11.2009)

PCT

(10) International Publication Number
WO 2009/139965 A2

(51) International Patent Classification:
G06F 17/30 (2006.01) *G06F 3/14* (2006.01)

(21) International Application Number:
PCT/US2009/038653

(22) International Filing Date:
27 March 2009 (27.03.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/121,485 15 May 2008 (15.05.2008) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, WA 98052-6399 (US).

(72) Inventors: **MURRAY, Michael, Charles**; One Microsoft Way, Redmond, WA 98052-6399 (US). **FLYNN, James, R.**; One Microsoft Way, Redmond, WA 98052-6399 (US). **WILLIAMS, Antony, Scott**; One Microsoft Way, Redmond, WA 98052-6399 (US). **MOLLICONE, Laurent**; One Microsoft Way, Redmond, WA 98052-6399 (US). **JAYADEVAN, Siddharth**; One Microsoft Way, Redmond, WA 98052-6399 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, 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, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, 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))*

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) **Title**: DATA VIEWER MANAGEMENT

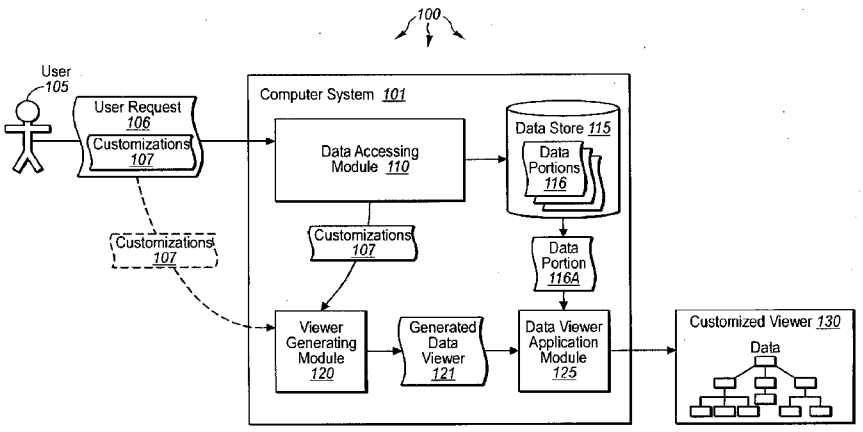

FIG. 1

(57) **Abstract**: Embodiments described herein are directed to generating a customized data viewer, where the viewer is configured to display data at any level in a data model. In one embodiment, a computer system receives a user request indicating that portions of data are to be displayed in a user-customized manner using a data viewer. The computer system accesses the requested data portions that are to be displayed with the data viewer. The computer system generates a dynamic data viewer configured to display the accessed data portions in the user-customized manner indicated in the received user request. The computer system also applies the generated dynamic data viewer to the accessed data portions, such that the generated viewer displays the requested data portions in the user-customized manner.

WO 2009/139965 A2

DATA VIEWER MANAGEMENT

BACKGROUND

[0001] Computers have become highly integrated in the workforce, in the home, in mobile devices, and many other places. Computers can process massive amounts of information quickly and efficiently. Software applications designed to run on computer systems allow users to perform a wide variety of functions including business applications, schoolwork, entertainment and more. Software applications are often designed to perform specific tasks, such as word processor applications for drafting documents, or email programs for sending, receiving and organizing email.

[0002] In many cases, software applications are designed to interact with other software applications or data on other computer systems. For example, some software applications may be designed to display data in the form of a model, or in some other form. Models may be used to illustrate objects along with various relationships between the objects. For example, a process flow diagram may illustrate the various steps involved in a particular process, and how each step relates to other steps (e.g., processing order, etc.).

[0003] Often, it is desirable to view data from different perspectives. For example, different users often have different needs with regard to the data. An end-user, for instance, may look at certain pieces of information from a high level, whereas a system architect may choose a more detailed perspective. In many cases, data viewing perspectives are specified in the application viewer. When a particular perspective is chosen to display a portion of content, that perspective is typically applied to the entire data portion in a rigid, uniform fashion.

BRIEF SUMMARY

[0004] Embodiments described herein are directed to generating a customized data viewer, where the viewer is configured to display data at any level in a data model. In one embodiment, a computer system receives a user request indicating that portions of data are to be displayed in a user-customized manner using a data viewer. The computer system accesses the requested data portions that are to be displayed with the data viewer. The computer system generates a dynamic data

viewer configured to display the accessed data portions in the user-customized manner indicated in the received user request. The computer system also applies the generated dynamic data viewer to the accessed data portions, such that the generated viewer displays the requested data portions in the user-customized manner.

[0005] In another embodiment, a computer system accesses data that is to be displayed using an initial data viewer, the initial data viewer displaying data of various types according to one or more stored data type mappings. The computer system receives user input indicating that stored data type mappings are to be changed and dynamically modifies the initial data viewer based on the received user input. The computer system assigns a name to the modified initial data viewer and stores the named data viewer in a viewer repository, such that other users can access and use the named data viewer.

[0006] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] To further clarify the above and other advantages and features of embodiments of the present invention, a more particular description of embodiments of the present invention will be rendered by reference to the appended drawings. It is appreciated that these drawings depict only typical embodiments of the invention and are therefore not to be considered limiting of its scope. The invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

[0008] Figure 1 illustrates a computer architecture in which embodiments of the present invention may operate including generating a customized data viewer, where the viewer is configured to display data at any level in a data model.

[0009] Figure 2 illustrates a computer architecture in which embodiments of the present invention may operate including dynamically applying a modified data viewer at any level in a data model.

[0010] Figure 3 illustrates a flowchart of an example method for generating a customized data viewer, where the viewer is configured to display data at any level in a data model.

[0011] Figure 4 illustrates a flowchart of an example method for dynamically applying a modified data viewer at any level in a data model.

DETAILED DESCRIPTION

[0012] Embodiments described herein are directed to generating a customized data viewer, where the viewer is configured to display data at any level in a data model. In one embodiment, a computer system receives a user request indicating that portions of data are to be displayed in a user-customized manner using a data viewer. The computer system accesses the requested data portions that are to be displayed with the data viewer. The computer system generates a dynamic data viewer configured to display the accessed data portions in the user-customized manner indicated in the received user request. The computer system also applies the generated dynamic data viewer to the accessed data portions, such that the generated viewer displays the requested data portions in the user-customized manner.

[0013] In another embodiment, a computer system accesses data that is to be displayed using an initial data viewer, the initial data viewer displaying data of various types according to one or more stored data type mappings. The computer system receives user input indicating that stored data type mappings are to be changed and dynamically modifies the initial data viewer based on the received user input including the mappings changes while the data is being presented. The computer system assigns a name to the modified initial data viewer and stores the named data viewer in a viewer repository, such that other users can access and use the named data viewer.

[0014] Embodiments of the present invention may comprise or utilize a special purpose or general-purpose computer including computer hardware, as discussed in

greater detail below. Embodiments within the scope of the present invention also include physical and other computer-readable media for carrying or storing computer-executable instructions and/or data structures. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer system. Computer-readable media that store computer-executable instructions are physical storage media. Computer-readable media that carry computer-executable instructions are transmission media. Thus, by way of example, and not limitation, embodiments of the invention can comprise at least two distinctly different kinds of computer-readable media: physical storage media and transmission media.

[0015] Physical storage media includes RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer.

[0016] A "network" is defined as one or more data links that enable the transport of electronic data between computer systems and/or modules and/or other electronic devices. When information is transferred or provided over a network or another communications connection (either hardwired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a transmission medium. Transmission media can include a network and/or data links which can be used to carry or transport desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. Combinations of the above should also be included within the scope of computer-readable media.

[0017] However, it should be understood, that upon reaching various computer system components, program code means in the form of computer-executable instructions or data structures can be transferred automatically from transmission media to physical storage media. For example, computer-executable instructions or data structures received over a network or data link can be buffered in RAM within a network interface card, and then eventually transferred to computer system RAM

and/or to less volatile physical storage media at a computer system. Thus, it should be understood that physical storage media can be included in computer system components that also (or even primarily) utilize transmission media.

[0018] Computer-executable instructions comprise, for example, instructions and data which cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, or even source code. Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the described features or acts described above. Rather, the described features and acts are disclosed as example forms of implementing the claims.

[0019] Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including, personal computers, desktop computers, laptop computers, message processors, hand-held devices, multi-processor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, pagers, routers, switches, and the like. The invention may also be practiced in distributed system environments where local and remote computer systems, which are linked (either by hardwired data links, wireless data links, or by a combination of hardwired and wireless data links) through a network, both perform tasks. In a distributed system environment, program modules may be located in both local and remote memory storage devices.

[0020] Figure 1 illustrates a computer architecture 100 in which the principles of the present invention may be employed. Computer architecture 100 includes computer system 101. In some embodiments, computer system 101 may include a variety of modules, data stores, and other components. Each is capable of intercommunication with the other components, and may be configured to communicate with modules and components of other computer systems connected

via a computer network. Computer system 101 includes data accessing module 110. In some embodiments, data accessing module 110 may be configured to receive user request 106 and customizations 107 from user 105. User 105 may be any type of computer user including an end-user, a software engineer, a system administrator, or other user.

[0021] User request 106 may include an indication that various data portions displayable in a software application are to be displayed in a customized manner. For example, a software application running on computer system 101 may output data that is displayed for user 105 on some type of display. The software application may be configured to access and edit certain types of information accessed in data store 115. The data (e.g., data portions 116) may include any type of information including files or file portions, raw data, searchable database information or any other type of data. User 105 may desire to view all or a portion of the data in a customized manner. User 105 may indicate in customizations 107 how the data is to be displayed.

[0022] For example, a software application may display data portions 116A as a flow chart with corresponding objects and connectors. User 105 may desire to view some of data portions 116A in a list view, or as an embedded spreadsheet. Additionally or alternatively, different background images, shapes or colors may be applied. Along those lines, similar or different changes may be made to any text included in data portions 116A. In some embodiments, customizations 107 may be received at data accessing module 110 and passed to viewer generating module 120. In other embodiments, customizations 107 may be sent directly to viewer generating module 120. After receiving user request 106 with customizations 107, data accessing module 110 may access data portion 116A based on user request 106. Data portion 116A may be sent to data viewer application module 125 for application to a generated viewer.

[0023] In some embodiments, viewer generation module 120 may be configured to dynamically generate a data viewer based on customizations 107. In some cases, module 120 may alter or modify an existing viewer. For instance, a viewer that is currently being used to display software application data may be dynamically

modified by module 120 according to customizations 107. Additionally or alternatively, module 120 may generate a data viewer (e.g., data viewer 121) and pass the viewer to data viewer application module 125. Module 125 may be configured to apply data viewer 121 to data portion 116. Thus, in some embodiment, viewer 121 generated based on customizations 107 may be used to view data portion 116 selected by user request 106. The resulting customized viewer 130 may thus display data portion 116A along with other data portions for viewing by user 105. In some embodiments, customized viewer 130 may be stored in a central database and made available to other users for application to other data portions. These and other concepts will be explained in greater detail with regard to method 300 of Figure 3 below.

[0024] Figure 3 illustrates a flowchart of a method 300 for generating a customized data viewer, where the viewer is configured to display data at any level in a data model. The method 300 will now be described with frequent reference to the components and data of environment 100.

[0025] Method 300 includes an act of receiving a user request indicating that one or more portions of data are to be displayed in a user-customized manner using a data viewer (act 310). For example, data accessing module 110 may receive user request 106 indicating that data portion 116A is to be displayed in a user-customized manner using generated data viewer 121. In some cases, data portion 116A may include data corresponding to a data model. For example, the data may be part of an organizational flowchart, or part of a system work flow model, or other portion of organized information. User 105 may desire to specify how certain portions of information are displayed. For instance, a user may specify that all text referring to a certain object is italicized. Or, a user may specify that all objects with a certain hierarchical relationship to a designated object are displayed in list form, or in embedded spreadsheet form, or in a certain color, font, size, shape, presented with a certain animation or transition, color scheme, background, or other customization. Many other customizations may be implemented in the system herein.

[0026] Method 300 includes an act of accessing the requested data portions that are to be displayed with the data viewer (act 320). For example, data accessing module 110 may access data portions 116/116A that are to be displayed with customized viewer 130. As mentioned above, the data in data store 115 may be any type of data, stored in any format or file configuration. Data accessing module 110 may access data in data store 115 or in any other data store in computer system 101 or in another computer system or storage network. Data portion 116A may include one or more portions of a file, one or more entire files or groups of files, or simply raw data which can be queried using database query commands.

[0027] Method 300 includes an act of generating a dynamic data viewer configured to display the accessed data portions in the user-customized manner indicated in the received user request (act 330). For example, viewer generating module 120 may generate dynamic data viewer 121 configured to display data portion 116A in the user-customized manner indicated in user request 106. In some embodiments, user request 106 may indicate which data is to be displayed and customizations 107 may indicate how the data is to be displayed in the viewer. Viewer generating module 120 may dynamically generate viewer 121 for data 116 while an application configured to edit or use data 116 is in operation. Thus, if an application is accessing and displaying data 116 on a default display, viewer generating module 120 may be configured to generate a new viewer while the application is running based on user request 106 and/or customizations 107.

[0028] In cases where a default or current viewer is being used, viewer generating module 120 may be configured to modify the default viewer according to the customizations indicated by user 105. In such cases, module 120 may initially copy the default viewer to data store 115 or to some other store for maintaining data viewers. Thus, the default viewer and other viewers may be archived and made available to other users. Similarly, any customized viewers generated by viewer generating module 120 (e.g., customized viewer 130) may be stored in a viewer store and made available to other computer system or computer network users. The network may include a local area network, an intranet, the internet, or any other type of network. In cases where viewer generating module 120 modifies

a current or default data viewer, some of the data requested in user request 106 may be displayed using the default viewer and the remainder may be displayed using the modified viewer.

[0029] Method 300 includes an act of applying the generated dynamic data viewer to the accessed data portions, such that the generated viewer displays the requested data portions in the user-customized manner (act 340). For example, data viewer application module 125 may apply generated data viewer 121 to data portion 116A, where the generated viewer displays the data portions requested in request 106 in the user-customized manner. Thus, different viewers may be generated and applied to different data portions, as requested by the user.

[0030] In some embodiments, where data portion 116 corresponds to model data, the model data may be displayed according to a current display view using a current data viewer. User input (e.g., user request 106) may be received indicating the view is to be switched to a different display view. The user input includes one or more data viewer edits that are to be applied to the current data viewer resulting in a customized data viewer. Computer system 101 may switch display views from the current display view to the modified display view by dynamically editing the current data viewer according to the received data viewer edits. The applied edits, in this case, may cause the display view to be switched from the current view to the modified view displayed with the edited, customized data viewer. In some scenarios, each of the different views represents a different perspective of the model data.

[0031] Additionally or alternatively, user 105 may use the generated dynamic data viewer to modify one or more other data viewers. Using a generated viewer, a viewer's data may be modified, resulting in another, different customized viewer. Thus, viewers may be combined, edited or otherwise modified to suit a particular user's needs. The user may customize multiple data viewers, and use each in different scenarios. Rules may also be established that dictate when a certain viewer is to be used. For example, user 105 may dictate that when certain data is being displayed, Customized Viewer A is to be used, and when other data is being displayed, Customized Viewer B is to be used. The transition between viewers

may occur instantaneously, once the new or other data is requested for display. Viewers may be generated or selected based on a user's artistic tastes, functionality needs or any other criteria. Any data subsequently accessed may be displayed according to a user's customized viewer, a default viewer, or by another user's customized viewer.

[0032] In one embodiment, data accessing module 110 may receive a user request indicating that among a plurality of data portions (e.g., 116) corresponding to a data model, each data type in the model is to be displayed in a user-customized manner according to the data type using a data viewer. Data accessing module 110 may access the requested data portions of the model that are to be displayed with the data viewer (e.g., viewer 121). Viewer generating module 120 may generate a dynamic data viewer (e.g., 121) configured to display each accessed data portion of the model according to the user-customized manner for the data type as indicated in the received user request. Thus, viewers may be generated in a different manner, depending on the data type indicated in the request. Moreover, data viewer application module 125 may apply the generated dynamic data viewer to the accessed data portions of each data type, where the generated viewer (e.g., customized viewer 130) displays each requested data portion in the user-customized manner according to the data type. Data viewer generation and modification will be explained in greater detail below with regard to computer architecture 200 of Figure 2 and method 400 of Figure 4.

[0033] Figure 2 illustrates a computer architecture 200 in which the principles of the present invention may be employed. Figure 4 illustrates a flowchart of a method 400 for dynamically applying a modified data viewer at any level in a data model. The method 400 will now be described with frequent reference to the components and data of environments 100 of Figure 1 and 200 of Figure 2.

[0034] Method 400 includes an act of accessing data that is to be displayed using an initial data viewer, the initial data viewer displaying data of various types according to one or more stored data type mappings (act 410). For example, data accessing module 110 may access data portion 116A that is to be displayed using initial data viewer 230, where viewer 230 displays data of various types according

to stored data type mappings. Accordingly, when certain data types are to be viewed in a predetermined manner, the data types would have mappings to certain viewers or views within a viewer. Thus, where a user (e.g., user 105) has selected a particular viewer for a model type, the appropriate viewer may be selected from stored data viewers 221 based on the established mapping.

[0035] In some embodiments, an appropriate viewer may be selected for a portion of model data based on the context of the data portion. For example, if data portion 116A is part of a list of data items, an appropriate viewer for that data may include a viewer that displays the data in a list form. Additionally or alternatively, if the data portion is part of a table or spreadsheet (or is embedded or nested in such), or is part of a collection or is a single item, an appropriate viewer for that data may be selected that is configured to display the data in an appropriate form, based on the data's context. As indicated above, user 105 may customize and specify, for each context, how the data is to be displayed. Different views may include table, list, flow diagram, master-detail (e.g., where the master is displayed as a tree and the details are displayed as a list), embedded, and other views.

[0036] Method 400 includes an act of presenting the accessed data using the initial data viewer (act 420). For example, initial data viewer 230 may present any or all of data portions 230A, 230B, 230C, 230D, 230E, 230F and 230G. Data objects 230A-G may be displayed as an object-link diagram as shown, or in some other form. The initial data viewer 230 may be a default viewer for a certain software application, or be the default viewer for a certain type of information. For example, default viewer 230 may be used whenever a database accessing software application is used, or whenever organizational workflows are to be displayed. Such associations between data viewers and data or data types may be referred to as data type mappings (e.g., 222).

[0037] Method 400 includes an act of receiving user input indicating that one or more stored data type mappings are to be changed (act 430). For example, data viewer modification module 210 may receive user input 205 indicating that stored data type mappings 222 are to be changed. In some cases, modifications made to mappings 222 may be user specific. That is, each user may make changes to

mappings 222 that only affect that user's mappings. Additionally or alternatively, settings may be configured to allow one user's changes to affect other user's mappings. Also, some mappings may be system-wide and may be permanent or only changeable by a system administrator.

[0038] Method 400 includes an act of dynamically modifying the initial data viewer based on the received user input including the mappings changes while the data is being presented (act 440). For example, data viewer modification module 210 may dynamically modify initial data viewer 230 based on received user input 205 including changes made to mappings 222 while the data is being presented in the initial viewer. This modification may result in modified data viewer 240. Such dynamic modification allows data to be displayed in initial viewer 230 and be modified at the same time. Thus, an initial data viewer may be used to modify the data of the initial data viewer. The initial data viewer (and other modified viewer) is comprised of editable data that can be edited on-the-fly while the viewer is being used to display other data portions.

[0039] Thus, in Figure 2, Modified data viewer 240 may include objects that have remained unchanged in the modification including 240A, 240B, 240C and 240E. Other data objects, however, have been modified such that they are displayed in a different, modified manner. For example, object 230D is now displayed as embedded spreadsheet 240D. Likewise, objects 240F and 240G are now displayed in modified data viewer 240 as list objects in a list. These modifications may be made as the result of a determination that the context of those data objects dictated a change. For instance, as explained above, if it is determined that the context for data object 230D is embedded, and the data is appropriately displayed as a spreadsheet, modification module 210 may make the appropriate modification according to mappings 222, resulting in embedded spreadsheet 240D. Similarly, if it is determined that the context for data objects 230F and 230G is combination (as opposed to single), and the data is appropriately displayed as items in a list, modification module 210 may make the appropriate modification according to mappings 222, resulting in List Item 1 (240F) and List Item 2 (240G).

[0040] Many other contexts and display forms are possible, each of which (including corresponding data type mappings) is fully customizable by the user. For instance, a user may desire to view a model from different perspectives such as a module perspective, parameter perspective, high-level, low-level and in-between. Moreover, a user may desire to have displayed a unified view with multiple perspectives displayed in single viewer. Thus, for example, business processes and architectural pieces may be displayed using the same viewer. Other users may wish to view the same data according to different perspectives to suit their needs. The user may thus modify or customize the viewer to display each type of information at any level in the model according to a user-defined manner.

[0041] Method 400 includes an act of assigning a name to the modified initial data viewer (act 450). For example, data viewer modification may receive data viewer name 206 and assign the received name to modified data viewer 240. In some cases, viewers may be combined, and one portion of a viewer may be used to display one portion of data, and another viewer may be used to display another portion of data. In some cases, user 105 may request that a stored viewer (e.g., data viewer 221 stored in data store 220) be used to display at least a portion of data requested in user request 106. In some cases, a previously stored viewer may be requested and used to view the entire portion of requested data.

[0042] Method 400 includes an act of storing the named data viewer in a viewer repository, such that other users can access and use the named data viewer (act 460). For example, modified data viewer 240 may be stored in data store 220. Data store 220 may be accessible to other computer (network) users which may request certain viewers for displaying different portions of data. In some cases, storing the named viewer in a viewer repository (e.g., data store 220) comprises storing only the changes made to initial viewer 230. Such an embodiment may reduce storage burdens, as only the modifications to the initial viewer are stored.

[0043] In some embodiments, the named modified data viewer 240 may be dynamically applied to the accessed data, such that the data is displayed according to the mappings associated with the modified viewer. In some cases, a user may access data store 220 and browse available viewers. The viewers (e.g., 221) may

include default or professionally designed viewers and may include user created views, which may include modified versions of the professionally designed viewers. These viewers, as explained above, may be configured to display data in a user-specified manner at any level of a data model, and may be used to edit the data models themselves. Data viewers may be selected for and used with a specific data type. In some embodiments, a user may be able to specify (e.g., in a drop down list) a data type and search for those data viewers that correspond to the data type. Other types of searching based on data context or data type may also be implemented.

[0044] The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

CLAIMS

We claim:

1. At a computer system in computer networking environment, a method for generating a customized data viewer (130), the viewer being configured to display data at any level in a data model, the method comprising:

an act of receiving a user request (106) indicating that one or more portions of data are to be displayed in a user-customized manner using a data viewer;

an act of accessing the requested data portions (116) that are to be displayed with the data viewer;

an act of generating a dynamic data viewer (130) configured to display the accessed data portions in the user-customized manner indicated in the received user request (106); and

an act of applying the generated dynamic data viewer (130) to the accessed data portions (116), such that the generated viewer displays the requested data portions in the user-customized manner.

2. The method of claim 1, wherein the generated customized data viewer is a modified version of a default data viewer.

3. The method of claim 2, further comprising an act of copying the default data viewer to a viewer repository.

4. The method of claim 2, wherein the default data viewer displays at least a portion of the requested data.

5. The method of claim 1, wherein the requested data corresponds to a data model.

6. The method of claim 5, further comprising:

an act of displaying the model according to a first display view;

an act of receiving user input indicating the view is to be switched to a second display view, the input comprising one or more data viewer edits that are to be applied to the customized data viewer; and

an act of switching display views from the first display view to the second display view by dynamically editing the data viewer according to the received data

viewer edits, the applied edits causing the display to be switched from the first view to the second view.

7. The method of claim 6, wherein each view represents a different perspective of the data model.

8. The method of claim 1, further comprising an act of the user using the generated dynamic data viewer to modify one or more other data viewers.

9. The method of claim 1, wherein any data subsequently accessed is displayed according to the user-customized view.

10. At a computer system in computer networking environment, a method for dynamically applying a modified data viewer at any level in a data model, the method comprising:

an act of accessing data that is to be displayed using an initial data viewer (230), the initial data viewer displaying data of various types according to one or more stored data type mappings (222);

an act of presenting the accessed data using the initial data viewer (230);

an act of receiving user input (205) indicating that one or more stored data type mappings (222) are to be changed;

an act of dynamically modifying the initial data viewer (230) based on the received user input (205) including the mappings changes while the data is being presented;

an act of assigning a name (206) to the modified initial data viewer (240); and

an act of storing the named data viewer (240) in a viewer repository (220), such that other users can access and use the named data viewer.

11. The method of claim 10, further comprising dynamically applying the named viewer to the accessed data, such that at least one portion of data is displayed according to the mappings.

12. The method of claim 10, wherein a second user accesses the repository and selects a customized viewer corresponding to a certain data type.

13. The method of claim 12, wherein only those viewers that correspond to the data type are available to the second user.

14. The method of claim 10, further comprising selecting an appropriate viewer for a model type based on the established mapping.

15. The method of claim 10, further comprising an act of selecting an appropriate viewer for at least a portion of model data based on the context of the data portion.

16. The method of claim 15, wherein the context of the data portion comprises at least one of: top-level, embedded, collection, single and item in a list.

17. The method of claim 10, wherein the act of storing the named viewer in a viewer repository comprises storing only the changes made to the initial viewer.

18. The method of claim 10, further comprising:

an act of receiving a user input requesting to use a stored named viewer to display at least a portion of requested data; and

an act of displaying the requested data using the stored named viewer.

19. A computer program product for implementing a method for creating a customized view that is substitutable for a standard model view at any level in the model, the computer program product comprising one or more computer-readable media having thereon computer-executable instructions that, when executed by one or more processors of the computing system, cause the computing system to perform the method, the method comprising:

an act of receiving a user request (106) indicating that among a plurality of data portions (116) corresponding to a data model, each data type in the model is to be displayed in a user-customized manner according to the data type using a data viewer;

an act of accessing the requested data portions of the model that are to be displayed with the data viewer;

an act of generating a dynamic data viewer (130) configured to display each accessed data portion of the model according to the user-customized manner for the data type as indicated in the received user request (106); and

an act of applying the generated dynamic data viewer (130) to the accessed data portions of each data type, such that the generated viewer displays each requested data portion in the user-customized manner according to the data type.

20. The method of claim 19, further comprising an act of the user using the generated dynamic data viewer to modify one or more other data viewers.

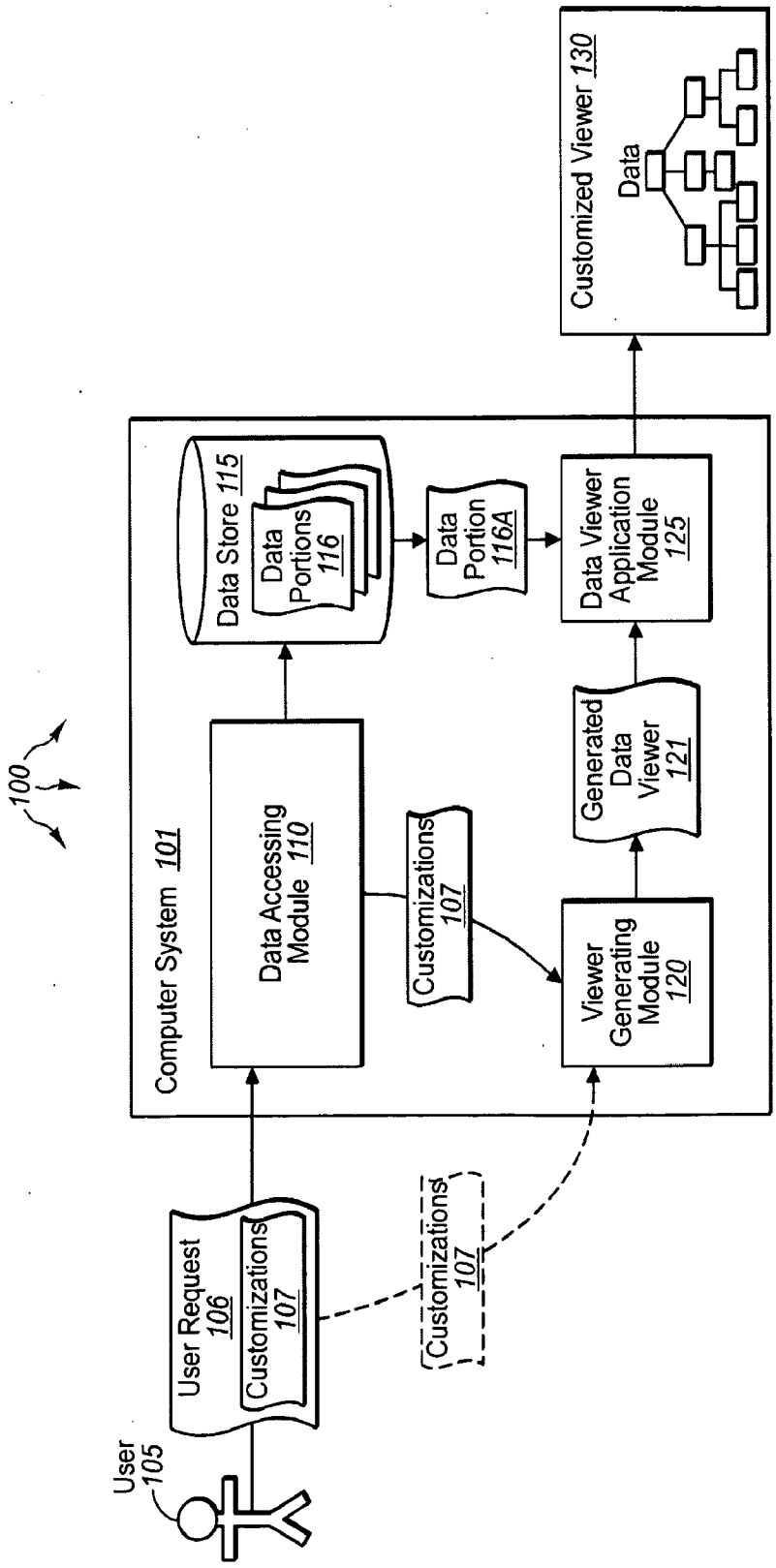
100
User
105
User Request
106
Customizations
107
Customizations
107
Computer System
101
Data Accessing
Module
110
Customizations
107
Data Store
115
Data
Portions
116
Data
Portion
116A
Viewer
Generating
Module
120
Generated
Data
Viewer
121
Data Viewer
Application
Module
125
Customized Viewer
130
Data

FIG. 1

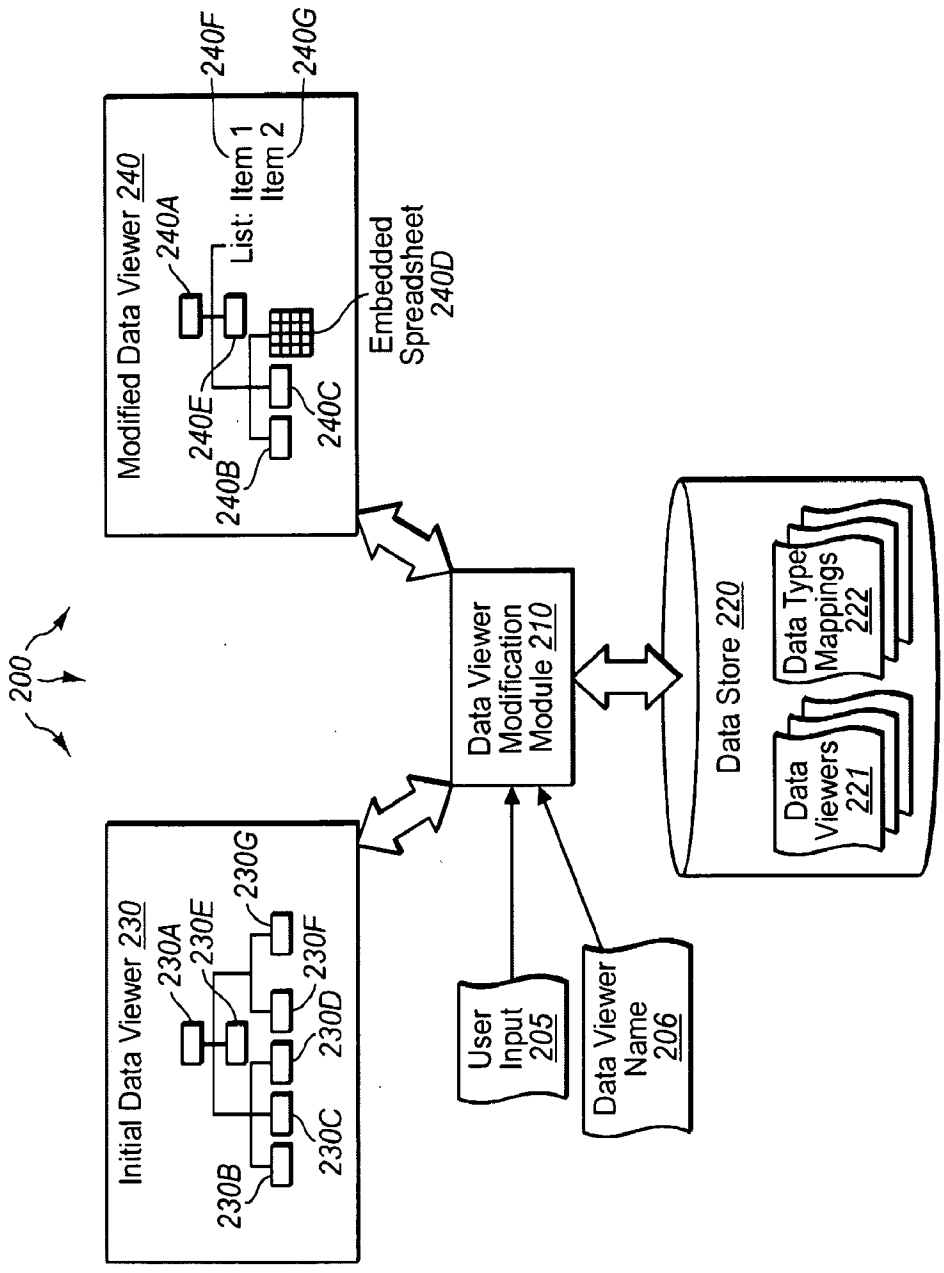
200
Initial Data Viewer 230
230A
230E
230G
230F
230D
230C
230B
Modified Data Viewer 240
240A
List: Item 1
Item 2
240F
240G
Embedded Spreadsheet 240D
240E
240C
240B
Data Viewer Modification Module 210
User Input 205
Data Viewer Name 206
Data Store 220
Data Viewers 221
Data Type Mappings 222

FIG. 2

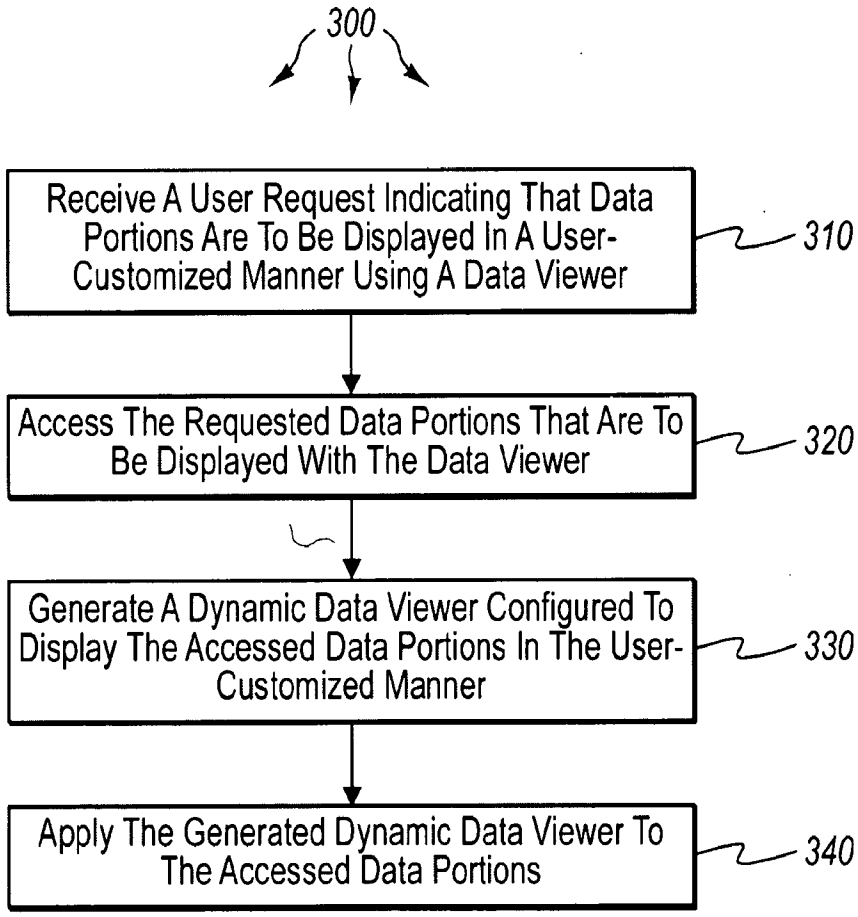
300
Receive A User Request Indicating That Data Portions Are To Be Displayed In A User-Customized Manner Using A Data Viewer
310
Access The Requested Data Portions That Are To Be Displayed With The Data Viewer
320
Generate A Dynamic Data Viewer Configured To Display The Accessed Data Portions In The User-Customized Manner
330
Apply The Generated Dynamic Data Viewer To The Accessed Data Portions
340

FIG. 3

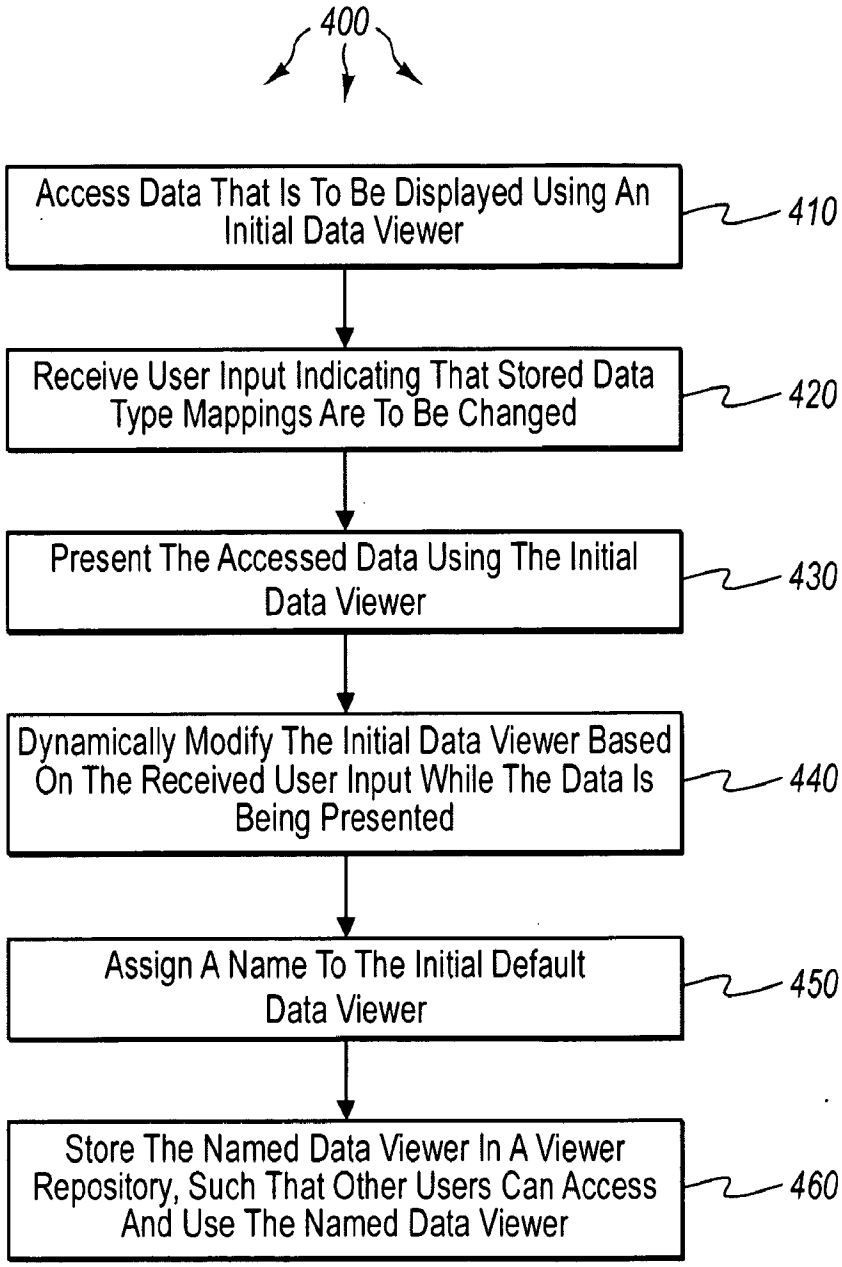
400
Access Data That Is To Be Displayed Using An Initial Data Viewer
410
Receive User Input Indicating That Stored Data Type Mappings Are To Be Changed
420
Present The Accessed Data Using The Initial Data Viewer
430
Dynamically Modify The Initial Data Viewer Based On The Received User Input While The Data Is Being Presented
440
Assign A Name To The Initial Default Data Viewer
450
Store The Named Data Viewer In A Viewer Repository, Such That Other Users Can Access And Use The Named Data Viewer
460

FIG. 4