(19) Octrooicentrum Nederland

(11) **2013306**

(12) A **OCTROOIAANVRAAG**

(21) Aanvraagnummer: **2013306**

(22) Aanvraag ingediend: **06.08.2014**

(51) Int.Cl.:
G06Q 10/00 (2012.01)

(30) Voorrang:
08.08.2013 US 61/863792
08.08.2013 US 61/863814
27.11.2013 US 61/909949
06.01.2014 US 14/148568

(41) Aanvraag ingeschreven:
10.02.2015

(43) Aanvraag gepubliceerd:
18.02.2015

(71) Aanvrager(s):
Palantir Technologies, Inc. te Palo Alto, California, Verenigde Staten van Amerika (US).

(72) Uitvinder(s):
Andy Isaacson te Palo Alto, Californië (US).

(74) Gemachtigde:
Ir. K.J. Metman te AMSTERDAM.

(54) **Template System For Custom Document Generation.**

(57) Man-machine interfaces (user interfaces) for generating documents and, more specifically, for receiving user selections of data objects and templates to generate documents are disclosed. In some embodiments, the man-machine interfaces for generating template-based documents may include fewer steps, result in faster creation of documents, consume less processing and/or memory resources, permit users to have less knowledge of programming languages and/or software development techniques, and/or allow less users or developers to create documents than the man-machine interfaces described above. In some embodiments, a user creates a mockup and/or placeholder document in an editor of their choice and generates a template by editing the mockup and/or placeholder document by inserting executable code instructions.

NL A 2013306

NL 21590-NH/mv

Template System For Custom Document Generation

Field

This invention relates to generating custom template-based documents.

Background

In many fields, computer programs may be written to programmatically generate reports or documents from the electronic collections of data. This approach requires a computer programmer to write a program to access electronic collections of data and output the desired report or document. Typically, a computer programmer must determine the proper format for the report or document from users or analysts that are familiar with the report or document. Some man-machine interfaces for generating reports or documents in this manner are software development tools that allow a computer programmer to write and test computer programs. Following development and testing of the computer program, the computer program must be released into a production environment for use. Thus, this approach for generating reports or documents may be inefficient because an entire software development life cycle (e.g., requirements gathering, development, testing, and release) may be required even if only one element or graphic of the report or document requires changing. Furthermore, this software development life cycle may be inefficient and consume significant processing and/or memory resources.

Summary

The systems, methods, techniques, and devices described herein each have several aspects, no single one of which is solely responsible for its desirable attributes. Without limiting the scope of this disclosure, several non-limiting features will now be discussed briefly.

Embodiments of the present disclosure relate to man-machine interfaces for generating documents and, more specifically, for man-machine interfaces for receiving user selections of data objects and templates to generate documents. In some embodiments, the man-machine interfaces for generating template-based documents may include fewer steps, result in faster creation of documents, consume less processing and/or memory resources, permit users to have less knowledge of programming languages and/or software development techniques, and/or allow less users or developers to create documents than the man-machine interfaces described above. Thus, in some embodiments, the man-machine interfaces described herein may be more efficient as compared to the man-machine interfaces described above. For

example, in some embodiments, a user creates a mockup and/or placeholder document in an editor of their choice and generates a template by editing the mockup and/or placeholder document by inserting executable code instructions. Typically, users have a preferred document editor that they are proficient and/or are comfortable with. The user may then select one or more data objects and the template, via one or more man-machine interfaces, which causes access of the one or more selected data objects through execution of the code instructions in the template to generate the report. The methods and systems described herein may advantageously reduce the steps and/or cognitive burden of a user interacting with the man machine-interfaces because reports and/or documents may be created according to certain specifications without the completion of full software development life cycles and/or the use of software development tools described above. Additionally, the man machine-interfaces described herein may use less processing and/or memory resources than the tools used for traditional software development because compilation and releases to production may not be necessary following changes to a report. Also, the systems and methods described herein may use less computational resources than traditional techniques for report generation because sections of the template document can be ignored by the system since only the executable code instructions in the template may need to be evaluated and/or executed.

In some embodiments, a computer system for generating custom template-based documents comprises a template storage device. The computer system may further comprise one or more hardware processors programmed via executable code instructions. When executed, the executable code instructions may cause the computer system to implement a template generator. The template generator may be configured to access a placeholder template comprising one or more placeholders indicating locations for insertion of executable instructions. The template generator may be further configured to receive executable instructions to be included in the placeholder template. The template generator may be further configured to store, in the template storage device, a template including one or more sets of the received executable instructions inserted into corresponding placeholders of the placeholder template. When further executed, the executable code instructions may cause the computer system to implement a template selection unit. The template selection unit may be configured to receive a selection of the template and selection of one or more data objects to include in a generated custom document based on properties of the one or more data objects. When further executed, the executable code instructions may cause the computer system to implement a template processor. The template processor may be configured to parse the one or more sets of executable instructions included in the template. The template processor may be further configured to execute each set of executable instructions, wherein at least some

sets of executable instructions include instructions to access properties of the selected data objects stored in one or more data sources. The template processor may be further configured to generate an output for each set of executable instructions. The template processor may be further configured to generate the custom document by replacing sets of executable instructions in the template with the output generated by execution of corresponding sets of executable instructions. At least some of the output include properties of the selected data objects and/or summary data regarding the properties of the selected data objects.

In some embodiments, a method for generating custom template-based documents may comprise receiving selection of a template, the template including one or more sets of executable instructions. The method may further comprise receiving selection of one or more data objects to access in response to executing the one or more sets of executable instructions. The method may further comprise executing, by a computer system having one or more computer processors and an electronic storage device, each set of executable instructions, wherein at least some sets of executable instructions include instructions to access properties of at least some of the selected data objects. The method may further comprise generating an output for each set of executable instructions. The method may further comprise generating a custom document by replacing sets of executable instructions in the template with the corresponding generated output, including properties of the selected data objects and/or summary data regarding the properties of the selected data objects.

In some embodiments, a non-transitory computer storage comprises instructions for causing a computer system to generate custom template-based documents. When executed, the instructions may receive a template that was modified by replacing placeholders with sets of executable instructions. When executed, the instructions may receive a selection of the template. When executed, the instructions may receive selection of one or more data objects to include in a generated custom document based on properties of the one or more data objects. When executed, the instructions may parse the one or more sets of executable instructions from the template by the computer system. When executed, the instructions may execute each set of executable instructions, wherein at least some sets of executable instructions include instructions to access properties of the selected data objects stored from one or more data sources. When executed, the instructions may generate an output for each set of executable instructions. When executed, the instructions may generate a custom document by replacing sets of executable instructions in the template with the corresponding generated output, including properties of the selected data objects and/or summary data regarding the properties of the selected data objects.

Advantageously, according to some embodiments, the disclosed techniques provide more effective and/or efficient man-machine interfaces for generating reports. A user may create a mockup and/or placeholder document in a document editor of their choice. The user may then use one or more man-machine interfaces to insert executable code instructions into the placeholder document. This approach for generating templates for documents may reduce the amount of time and effort of the user because the user may use document editors and document creation techniques they are familiar with, and documents may be created dynamically and/or flexibly without the need to release new versions of software. Users may select data objects and a template through one or more graphical user interfaces to dynamically generate a report from the data objects. Thus, the man-machine interfaces for selecting data objects and templates may reduce the cognitive burden of users, reduce the number of steps to create documents, and/or reduce the computational and/or memory resources for creating documents as compared to traditional programmatic approaches for generating documents.

This application is related to but does not claim priority from U.S. Patent No. 8,489,623 entitled "Creating Data In A Data Store Using A Dynamic Ontology" filed May 12, 2011, which is hereby incorporated by reference in its entirety and referred to herein as the "Ontology reference."

Brief Description of the Drawings

Certain aspects of the disclosure will become more readily appreciated as those aspects become better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings.

Figure 1 is a block diagram illustrating an example template system, according to some embodiments of the present disclosure.

Figure 2 is a flowchart illustrating an example document generation process from a template, according to some embodiments of the present disclosure.

Figure 3A illustrates an example placeholder template, according to some embodiments of the present disclosure.

Figure 3B illustrates an example Extensible Markup Language document of the placeholder template, according to some embodiments of the present disclosure.

Figure 3C illustrates an example Extensible Markup Language document of a template with executable code instructions, according to some embodiments of the present disclosure.

Figure 3D illustrates an example Extensible Markup Language document of an output document following the execution of embedded code instructions, according to some embodiments of the present disclosure.

Figure 3E illustrates an example custom document, according to some embodiments of the present disclosure.

Figure 4A illustrates an example user interface of the template system for loading and/or viewing data objects, according to some embodiments of the present disclosure.

Figure 4B illustrates an example user interface of the template system for selecting a template, according to some embodiments of the present disclosure.

Figure 4C illustrates an example user interface of the template system for evaluating code instructions, according to some embodiments of the present disclosure.

Figure 5 is a block diagram illustrating an example template system with which various methods and systems discussed herein may be implemented.

Detailed Description

Reports and/or documents may be programmatically generated from structured data. For example, a computer system may be programmed with code instructions to uniformly generate reports and/or documents from structured data. To modify the format of a report and/or document, a new software release may be required to make changes to the report and/or document. In another example, a report and/or document may be generated from the embedded features of a document processing application. Microsoft Word may have built in functions for executing code instructions in a document such as macros for executing Visual Basic.

In addition to computer systems programmed to generate uniform reports and/or the embedded features of a document processing application, disclosed herein are systems, methods, techniques, and devices for dynamically generating custom documents that include information related to one or more data objects and/or properties of those data objects. Using the techniques and systems described herein, computational and/or memory efficient,

flexible, scalable, and custom document generation may be achieved. Furthermore, the man machine-interfaces described herein may advantageously reduce the steps and/or cognitive burden of a user to customize reports based on data objects without the need of additional software releases. In one particular example implementation of the systems and methods discussed herein, custom documents may be generated based on properties of one or more user-selected data objects and based on Microsoft Word templates that include embedded executable code instructions that may be evaluated outside of Microsoft Word. While certain examples herein refer to Microsoft Word, the systems and methods are applicable to any data format and any reading and/or editing software.

Example Template Generation

Figure 1 illustrates a template system, according to some embodiments of the present disclosure. In the example embodiment of Figure 1, the template environment 190 comprises a network 160, a template system 100, a user computing device 102, and an object storage device 130. Various communications between these devices are illustrated. For example, communication of a placeholder template 110, data selection 120, object data 140, and a custom document 150 are illustrated in various actions 1-5 that are illustrated in the circled numbers in the Figure. In this embodiment, the template system 100 includes a template generator 104, a template storage device 106, a selection unit 108, and a template processor 109, each of which is described in further detail below.

As shown in Figure 1, at action one, a template with code placeholders, which is referred to herein as a "placeholder template," is transmitted from the user computing device 102 to the template generator 104 of the template system 100. In some embodiments, the placeholder template 110 includes code placeholders, which indicate locations in a template document where the user would like content to be added dynamically based on the properties of one or more selected data objects. Figure 3A, for example, illustrates an example placeholder template 300, with placeholders that have been added and/or displayed through a document processing application. The placeholder template 300 may correspond to the placeholder template 110 of Figure 1. A human operator may have created the placeholder template 300 in a document processing application, such as, but not limited to, Microsoft Word. The placeholder template 300 may comprise static text and/or features that may be used in all of the reports and/or documents generated from templates based on the placeholder template 300, such as static text 312. In other examples, the header or footer of the document may contain the same information on all reports, such as a company logo.

The human operator may have created the placeholder template 300 to meet specifications for a custom report. The layout may be customized to the specifications of the report and/or to the preferences of the human operator. For example, the title 304 and/or the table 302, may be aligned, formatted, and/or in a font that matches the human operator's preferences. Similarly, the image 306 may be positioned and/or aligned according to the human operator's preferences.

Elements of the placeholder template 300, such as the title 304, the image 306, and/or the table 302 may comprise placeholder elements. The placeholder elements represent aspects and/or features of the document that the human operator desires to be dynamically populated and/or updated with properties from data objects and/or based on the properties of data objects. For example, title-holder 304 may be a placeholder that is to be replaced with a title associated with an object selected by the requester of a document based on the template. For example, the title-holder 304 may be replaced with the name of a person, which may correspond to one or more name properties associated with a person data object. Similarly, the other placeholder elements, such as the image 306 and the height-holder, contact-holder, and associates-holder in table 302 should correspond to properties of one or more objects (e.g., person objects) selected by the requesting human operator.

A template may be in various formats. For example, a template may comprise documents and/or formats such as, but not limited to, Microsoft Word, Microsoft PowerPoint, Microsoft Excel, Hyper Text Markup Language ("HTML"), a database format, Extensible Markup Language ("XML"), JSON, delimited file formats, a file format that is proprietary to the template system 100, and/or any other format. Figure 3B illustrates, for example, an XML document of a placeholder template in an XML based document format. For example, the Microsoft Office document formats, such as, but not limited to, Microsoft Word, PowerPoint, Excel, any other document format, may comprise XML based document formats. In other words, for example, a Word document may be an archive file, corresponding to, but not limited to, a ZIP format, of XML documents. As such, a human operator, with the use of an archival application, or some other application, may open a document file archive to view and/or edit the one or more XML documents comprising the document.

The example XML document 310A may correspond to the placeholder template 300 of Figure 3A, which may be in a document archive format comprising one or more XML files, including the XML document 301A. The example tags in the XML document 310A are illustrative and may not correspond to a specific document XML format.

As illustrated by the XML document 310A, a human operator, upon opening the XML document 310A with a text editing application, XML editing application, and/or some other application, may identify the placeholder elements. For example, "<text>NAME-HOLDER</text>" may be identifiable because of the "HOLDER" text. Thus, a human operator may search through one or more documents 310 that comprise the template (e.g., there may be multiple XML and/or other document types that are part of a single Microsoft Word document) for the particular character string used in the template to identify placeholders. In the example of Figure 3, the human operator can search for the character string "HOLDER" to easily identify files and/or locations in the files where placeholders are present and where executable code should be inserted in order to make those placeholders operable to obtain data associated with one or more objects. Thus, the use of placeholders may be useful because the various files that make-up a single output format (e.g., Microsoft Word document) may be very large, complex, and/or otherwise difficult to navigate in order to identify the appropriate locations to replace placeholders with executable code. For example, a single XML document may be thousands of lines with hundreds of different tags. Thus, identifying portions of the XML document may be achieved by searching for placeholders in the XML document. Furthermore, a human operator may edit and/or modify the XML document without fully understanding a complex XML based document format because the placeholders may focus the human operator on the important sections of the XML for editing and/or modifying.

While XML is shown in the document 310A, the template system may support other previously mentioned document formats and/or any other document format. For example, a HTML template may be used, and the placeholders may be placed in a HTML document.

Returning to Figure 1, at action two, the template generator 104 generates a template by replacing the placeholders in the placeholder template 110 (or placeholder template 300 of Figure 3A) with code segments that are executable in order to obtain replacement data for those placeholders. In some embodiments, some or all of the embedded code instructions associated with placeholders may be placed and/or included in the placeholder template 110 that is sent from the user computing device 102 to the template generator 104 through the network 160. For example, the user 102 may write the code segments based on using a programming interface provided by the template generator 104, which may be implemented on the user computing device 102 partially and/or fully in various embodiments. Thus, the template generator 104 may be configured to receive executable instructions to be included in a document template, either by the user 102 and/or by another user that may have more programming skills that are useful in writing and/or selecting code associated with each

placeholder. For example, the template generator 104 may be a text editing application operated by a human operator that inserts the embedded code instructions based on the code placeholders. In some embodiments, the template generator 104 may be an automated process that receives the template 110 with code placeholders and automatically replaces the code instruction placeholders with embedded instructions. In some embodiments, template generation is automatic, manual, or some combination thereof. The template generator 104 may output and/or store the generated template with embedded code instructions in the template storage device 106.

Figure 3C illustrates the template with executable code instructions inserted, which may be distinguished from the placeholder template (e.g., Figure 3B). The XML document 310A of Figure 3B may have been modified to produce the coded XML document 310B. The executable code instructions in the coded XML document 310 are illustrative and, thus, may not correspond to any specific programming language or include instructions that are syntactically complete and/or correct. The template system may be configured to support one or more interpreted programming languages, such as, but not limited to, embedded Ruby, JRuby, Groovy, BASIC, Perl, Python, Jython, and/or LISP. The template system may also be configured to support other programming languages, such as, but not limited to, Java, Lua, C, C++, and/or C#.

The use of embedded code instructions in the coded XML document 310B may allow for dynamic document creation based on data objects. For example, the code instruction, "<% print(object.getName) %>," may cause the template system 100 of Figure 1 (and/or the template processor 109 discussed further below) to retrieve the name property of a selected object (or objects) and print that name property to the document. As previously mentioned, the human operator and/or a template generator may know where to insert the particular code instruction within that XML element because of the "NAME-HOLDER" indicator from the XML document 310A in Figure 3B. Similarly, the human operator and/or template generator may add embedded code instructions accessing data object properties based on the placeholders in the template corresponding to the image 306, and/or the table 302 of Figure 3A.

In some embodiments, there may be some variations of how the embedded code instructions are executed. For example, to execute interpreted programming languages, such as, but not limited to, Ruby and/or Groovy, the template system 100 of Figure 1 (and/or the template processor 109 discussed further below) may execute the embedded code instructions at runtime without compiling the embedded code instructions. In some embodiments, to

execute compiled or partially compiled programming languages, such as, but not limited to, Java and/or C++, the template system 100 of Figure 1 (and/or the template processor 109 discussed further below) may comprise a compiler unit that compiles the extracted code instructions to be able to execute those code instructions.

In some embodiments, the template generator 104 and/or template system 100 of Figure 1 may comprise tools and/or applications for editing XML based document formats and/or other document formats. For example, an XML based document formatting application may allow a human operator to open an XML based document format and edit the XML files of the XML based document format directly. As previously illustrated, an XML based document format may comprise a file archive of XML documents. Thus, without an XML based document formatting application and/or tool, a human operator may have to open the XML based document file archive before editing the XML files and/or re-archive the XML files after making the XML changes. An XML based document formatting application and/or tool may allow the human operator to easily open, edit, and/or save XML documents within an XML based document format by performing the archiving and/or re-archiving steps automatically.

The use of embedded code instructions in the document 310B may allow for custom programming logic for document generation. For example, the code instruction 312, "if object.getPhoneNumber is not null," includes an if-statement. Therefore, the following code instruction 314, "println('Phone Number: ' + object.getPhoneNumber)," may only be executed if the if-statement 312 evaluates to the boolean True. Thus, programming logic in the document 310B may allow for conditional logic based on data object properties. The code instruction block 316 may illustrate further programming logic. For example, a person object may have multiple records associated with the person, each record may have a height property, and, therefore, a person may have multiple height properties. However, it may be desirable to display the most common height of a person. Therefore, the code instruction "height = frequencyMap(records, 'height').getMostCommon" at 316 may determine the most common height property that may be printed to the document.

The use of embedded code instructions in the document 310B may allow for efficient custom document creation. For example, the first and last illustrated code instructions, respectively, "<% for each object in objects %>" and "<% end %>," which may be illustrative of a for loop, may cause the template processor 109, when executing the code, to perform operations within the loop for each of the one or more data objects. The use of a for loop may be efficient because the two lines of embedded code instructions will cause the template system to loop

over any number of objects and repeat the static text and/or executable code instructions within the for loop. For example, the for loop may enclose the respective elements corresponding to the title 304, image 306, and/or the table 302 of Figure 3A. Therefore, upon executing the embedded code instructions within template, a page and/or section will be generated per data object (each corresponding to one loop of the for loop). For example, if ten person objects were selected, determined, accessed, and/or loaded and sent to the template system, ten pages and/or sections each corresponding to one person object may be generated.

Example Selection and Execution of a Template

Returning to Figure 1, actions 1-2 describe operations that may be performed in generating a coded template. Once a coded template is generated and stored in the template storage device 106, that coded template can be selected by the human operator and/or any other users for use in creating a document based on the template. Actions 3-5 of Figure 1 illustrate example processes that may be performed by a user in selecting a template, selecting one or more objects from which information is to be used in the document generated based on the template, and generating that document based on the object properties.

In action 3, the user of the user computing device 102 selects one or more objects to be included in the document/report. For example, if the user is a law enforcement officer and once a report including information on each of 10 suspects, those 10 suspects may be selected. Referring to Figure 4A, for example, a user interface 400 of a software application configured to provide data regarding objects and to allow the user to select one or more of the data objects. The example user interface 400 comprises a search box 402, an object display area 404, and menu bar 406. A human operator, e.g., the user of the user computing device 102, by typing and/or entering data into the search box 402 may load, lookup, and/or retrieve one or more data objects. For example, by typing the name of a person, such as "John Doe," the person object 410 may be displayed in the object display area 404. The other person objects 412 (including objects 412A, 412B, and/or 412C) may be displayed automatically and/or after user interaction by the human operator with the person object 410. For example, a human operator may select the person object 410 and select an option to display associates and/or persons related to the person object 410. The links 414A, 414B, and/or 414C may display relationships between the person object 410 and related person objects 412A, 412B, and/or 412C, respectively. For example, the person objects 412 may be related to the person object 410, such as, but not limited to, associates, acquaintances, and/or family members. The user interface 400 may be capable of displaying any type of data object and/or may not be limited to displaying person data objects.

In some embodiments, the embedded code instructions may determine one or more additional data objects based on the one or more selected objects. For example, known associate objects and/or the arrest record objects for a selected person object may be determined programmatically from the embedded code instructions, such as by using techniques discussed in U.S. Patent Application Serial No. 13/968,265, and U.S. Patent Application Serial No. 13/968,123, which are hereby incorporated by reference in their entireties. For example, the selected person data object may be a starting point to determine one or more additional related data objects. For example, a stolen vehicle report object, which has color property of "red," may be selected by the human operator, and the embedded code instructions of a template may identify data objects associated with pictures of red cars observed speeding nearby the location of the stolen vehicle (e.g., as indicated in object properties of those data objects).

In some embodiments, the embedded code instructions may access data from other servers and/or websites, either internally and/or externally from the template system. For example, a Google Street View image may be access and/or loaded by the embedded code instructions. The Google Street View image may be associated with the location property of a data object. Other data may be accessed and/or loaded through the execution of code instructions by the template system 100 of Figure 1 (and/or the template processor 109 discussed further below), such as, but not limited to, currency exchange rates, weather data, new reports, and/or any other available information.

In addition to visually searching and/or showing data objects and/or relationships between data objects, the user interface 400 may allow various other manipulations. For example, data objects may be inspected (e.g., by viewing properties and/or associated data of the data objects), filtered (e.g., narrowing the universe of objects into sets and subsets by properties or relationships), and statistically aggregated (e.g., numerically summarized based on summarization criteria), among other operations and visualizations.

When a user has the appropriate objects selected for use in the generated document, the user may then select one or more of several available templates to be used in generating a document including properties of the selected objects. For example, Figure 4B illustrates the user interface 400 of Figure 4A, with a particular template selected for use in generating a document based on the selected objects (all of the objects displayed in this embodiment). In this example, the human operator has selected the template "MyTemplate" 426 by accessing an "INVESTIGATION" menu 420 selected, selecting the "Data Export" menu item 422, and

then choosing the template 426 from a list 424 of any templates available to the user. In some embodiments, by selecting a particular template (e.g., "MyTemplate" 426), the process of generating a custom document based on all of the data objects displayed in the objects in the display area 404 is initiated. In some embodiments, the template user interface 400 may allow the human operator to individually and/or in a group select data objects, whether or not those objects remain viewable in the display area 404. Thus, upon selecting the template menu item 426 a custom document may be generated from the selected data objects.

In some embodiments, the template system may have an interface that accepts the transmission and/or upload templates such that the templates may be present in the menu list 424. For example, after the creation and/or generation of a template with embedded code instructions, which may correspond to an XML based document format comprising the XML document 310B in Figure 3C, a human operator may then upload the template to the template system. The menu list 424 of the user interface may then be automatically populated with the latest list of available templates.

Returning to Figure 1, once the template and objects to be included in the generated document are selected, indications of those selections are transmitted to the template system 100 for use by the selection unit 108 in obtaining properties of the selected objects for use in the generated document and accessing the selected template. For example, the data selections 120 may be sent from the user computing device 102 to the selection unit 108 through the network 160. As noted above, the data selection 120 may comprise one or more data object selections and/or template selections. The selection unit 108 may load and/or retrieve the template from the template storage device 106 based on the data selection 120. The selection unit 108 may request, receive, and/or load information and/or data regarding the data objects identified in the selection 120 from the object storage device 130. The selection unit 108 may send the template and/or the data objects to the template processor 109.

In action 5, the template processor 109 generates the custom document 150 based on the template with embedded executable instructions and the data objects 140 (and/or properties of data objects) accessed at the object storage device 130. The custom document 150 with properties from the data objects outputted from the template processor 109 may be sent to the user computing device 120 through the network 160. Thus, the template processor 109 is configured to identify executable code in the template, access properties of the selected, determined, and/or loaded objects based on the executable code, and replace the executable code with the output of the particular executable code segment. Figure 3D illustrates the

example XML document of Figure 3C with the embedded code instructions replaced with information regarding selected data objects. The output XML document 320 may be configured for interpretation by viewing software, such as a word processor (e.g., Microsoft Word), a browser, and/or other software, in order to depict a custom document that includes the object data.

In this example, the element "<text>John Doe</text>" illustrates that what was originally the TITLE-HOLDER of the placeholder template has been replaced with the name property value of a person object that was accessed based on execution of the executable code "<%= print(object.getName) %>" in the template. Similarly, an image name corresponding to the John Doe person object is included in the output document XML 320 in place of the Image placeholder. The other HOLDER elements of the table 302 have also been replaced with properties of the selected object, in response to execution of the corresponding code segments that replaced the placeholders by the template processor 109.

While the output document XML 320 illustrates output from executing embedded instructions from a single person data object, the ellipsis 322 illustrates that multiple person data objects may have been accessed and corresponding embedded code instructions may have been executed. As a result, the ellipsis 322 may comprise multiple sections and/or pages of the output document that may correspond to multiple person data objects. For example, if the object selection 120 includes the four person objects identified in Figure 4A, the output document XML 320 may include separate sections (e.g., corresponding to separate sections of a document, pages of the document, or separate documents) for each of the selected objects.

Figure 3E illustrates an example custom document that may be returned to a user, such as the custom document 150 of Figure 1. In one embodiment, the custom document 330 is generated by the template processor 109, which is discussed further with reference to block 210 of Figure 2.The example custom document 330 illustrates how an output custom document may be perceived by a human operator viewing the custom document 330 in a document processing application (e.g., the user of the user computing device 102 viewing the document in a word processor). For example, the title 332, image 334, and/or the table 336 correspond to the XML elements of the custom document XML 320 of Figure 3D. As illustrated, the custom document 330 comprises multiple pages, which may correspond to multiple person objects being processed by the template system, such as the four example objects selected in the user interface of Figure 4A.

The template system may efficiently output the custom document 330 because it may be based on a template. For example, the template system may execute and/or replace the embedded code instructions of the template document as shown by the modified XML document 310B. Thus, the template system may reuse all of the static text and/or everything but the embedded instructions of the template document to generate the custom document. Furthermore, the template system may be agnostic as to the specific details of a particular document format, such as, but not limited to, Microsoft Word. Similar to a template that may comprise a XML based document format, the custom document 330 may also comprise an XML based document format. Thus, the custom document 330 may comprise the custom document XML 320 of Figure 3D.

The template system may output the custom document 330 populated with properties from data objects and/or matching the document format, editing, layout, etc. matching the template document 300 from Figure 3A the human operator may have originally created.

In some embodiments, the user computing device 102, the template system 100, and the object storage device 130 may be on the same computing device or multiple computing devices. In some embodiments, communication between the user computing device 102, the template system 100, and object storage device 130 may occur without the use of the network 160. For example, if the user computing device 102, the template system 100, and object storage device 130 were on the same computing device, communication may occur without the use of a network.

Data Objects

In some embodiments, data is conceptually structured according to an object data model represented by an ontology. The conceptual data model may be independent of any particular and/or specific type of data store. For example, each object of the conceptual data model may correspond to one or more rows in a relational database and/or an object in an in-memory cache.

In some embodiments, an ontology, as noted above, may include stored information providing a data model for storage of data in a data store. The ontology may be defined by one or more object types, which may each be associated with one or more property types. At the highest level of abstraction, a data object is a container for information representing things in the physical world. For example, a data object may represent an entity such as a person, a place, an organization, a market instrument, and/or some other noun. Data objects may represent an event that happens at a point in time and/or for a duration. Data objects

may represent a document and/or other unstructured data source such as an e-mail message, a news report, a written paper, and/or a written article. Each data object may be associated with a unique identifier that uniquely identifies the data object within the data store.

Different types of data objects may have different property types. For example, a "Person" data object might have an "Eye Color" property type and an "Event" data object might have a "Date" property type. Each property as represented by data in the data store may have a property type defined by the ontology used by the data store.

Objects may be instantiated in the data store in accordance with the corresponding object definition for the particular object in the ontology. For example, a specific monetary payment (e.g., an object of type "event") of US$30.00 (e.g., a property of type "currency") taking place on 3/27/2009 (e.g., a property of type "date") may be stored in the data store as an event object with associated currency and date properties as defined within the ontology.

The data objects defined in the ontology may support property multiplicity. For example, a data object may be allowed to have more than one property of the same property type. For example, a "Person" data object might have multiple "Address" properties or multiple "Name" properties.

In some embodiments, the data objects the template system receives may correspond to an ontology according to the systems, methods, and/or techniques disclosed in the Ontology reference.

Example Document Generation Process

Figure 2 is a flowchart illustrating a document generation process from a template, according to some embodiments of the present disclosure. The method of Figure 2 may be performed by the template system 100 of Figure 1, such as the various components of Figure 1 that are discussed above, including the template processor 109. Depending on the embodiment, the method of Figure 2 may include fewer or additional blocks and/or the blocks may be performed in order different than is illustrated.

Beginning at block 202, properties of one or more data objects are accessed. For example, a person object, including some or all of the properties of that person object, may be received by the template system 100. The person object may possess one or more properties, such as, one or more, names, addresses, and other data.

At block 204, a template is received and/or accessed. The template may correspond to one or more of the previously illustrated formats. For example, the template may be a Word Document.

At block 206, executable code instructions are parsed from the template. One or more escape characters may be used to indicate an executable code instruction block in the template. For example, the "<%" and "%>" characters may respectively indicate the beginning and end of an executable code instruction block and/or a set of executable code instructions. Other escape characters and/or tags may be used to indicate executable code instructions in a template. The executable code instructions may correspond to one or more programming languages.

At block 208, the executable code instructions from the template are executed. Data objects and/or properties of the data objects may be accessed by the executable code instructions. The executable code instructions may also contain programming logic. For example, executable code instructions may access a person object, check for properties such as residences, and/or only print the most recent resident address.

At block 210, a custom document is generated. The custom document may include properties of the received data objects and/or summary data regarding the properties of the received data objects. The document may correspond to the same format and/or type as the format and/or type of the template. For example, if the template was a Word Document, the document may also be a Word Document. The document may be generated by replacing the executable code instructions in the template with the corresponding generated output. The actual template document may not be modified. For example, a copy of the template document may be made, and the executable code instructions in the copy template may be replaced.

Making Changes to a Complete Template

A human operator may desire, want, and/or need to modify the look, design, formatting, and/or layout of a complete template after the embedded instructions have been added to the template. For example, the human operator may modify the original placeholder template, which did not contain any of the embedded code instructions, in a word processing application, such as, but not limited to, Microsoft Word. The human operator, or an automated process and/or tool, may then copy the embedded instructions from the existing template to the new placeholder template. As a result, a new template with the design, formatting, and/or layout changes in the updated document format, but still including the

proper embedded code instructions, is created. Thus, a human operator may make the necessary changes in the word processing application, which may alleviate the need for the human operator to understand and/or have to make changes to a complex document format, such as, but not limited to, an XML based document format.

Example User Interfaces

Figures 4A, 4B, and 4C illustrate example user interfaces of the template system, or a subset thereof, according to some embodiments of the present disclosure. In some embodiments, the user interfaces described above and below may be displayed in any suitable computer system and/or application, for example, in a web browser window and/or a standalone software application, among others. Additionally, the functionality and/or user interfaces of the system as shown in Figures 4A, 4B, and/or 4C may be implemented in one or more computer processors and/or computing devices, as is described with reference to Figure 5. As noted above, in some embodiments the user interface 400 may be used for the object and data selection 120 illustrated in Figure 1. In some embodiments, upon selecting a specific template menu item 426, a custom document including object data associated with the selected objects, is generated. As noted above, Figure 4A illustrate the user interface 400 displaying several objects 410, 412, and associations 414 between the objects. Depending on the software application (e.g., Palantir's Gotham software), objects may be selected in various manners. Also as discussed above, Figure 4B illustrates example menu options that may be used to select a particular template to be used in generating an output document including information regarding the selected objects.

Moving to Figure 4C, the example user interface 400 comprises an evaluation tool 430, which includes an input box 432, an execute button 434, and/or an output box 436. A human operator may use the evaluation tool 430 to evaluate and/or test executable code instructions to view their sample print output before embedding those and/or similar instructions in a template.

An example use case and/or scenario for the evaluation tool 430 may be the following. As previously illustrated, a human operator may load and/or retrieve data objects 410 and/or 412, and then select none, one, some, or all of the data objects. A human operator may then type and/or insert code instructions into the input box 432. For example, some or all of the code instructions from Figure 3C and/or any other code instructions may be entered into the input box 432. A human operator may then click, tap, and/or touch the execute button 432, which may cause the template system to execute the code instructions in the input box 432. In some embodiments, after the template system executes the code instructions, the output

box may be automatically populated with the output of the executed code instructions. For example, if the code instructions in the input box 432 were "<%= print(object.getName) %>," the output box 436 might display "John Doe," which may correspond to the name property of a selected data object. In some embodiments, the template processor 109 of Figure 1 may execute the code instructions from the input box 432.

In some embodiments, a human operator may use the evaluation tool 430 to generate a template document efficiently. For example, a human operator may test, experiment, and/or preview code instructions by seeing what the output of those code instructions might look like before embedding those instructions in a template. The evaluation tool 430 may allow the human operator to fix issues and/or problems with the code instructions before uploading a template into the template system. Without the evaluation tool 430, a human operator would need to generate a template, upload it to the template system, and then execute the template on data objects to see what the output of those code instructions would look like. Thus, the evaluation tool 430 may save the human operator time and/or reduce the time it takes to generate a template.

Snippets

In some embodiments, the template system may be used to insert snippets into a custom document according to the systems, methods, and/or techniques disclosed in the Snippet references. For example, the embedded code instructions of a template may be evaluated in order to import into the document data related to snippet objects, such as importing the text portion of the returned snippets into the report body, and/or the citation portion of the snippet into a footnote. In this way, each snippet may be automatically added to the report (and future reports based on the same template) automatically without human intervention.

Implementation Mechanisms

The various computing device(s) discussed herein, such as the template system 100 of Figure 1, are generally controlled and coordinated by operating system software, such as, but not limited to, iOS, Android, Chrome OS, Windows XP, Windows Vista, Windows 7, Windows 8, Windows Server, Windows CE, Unix, Linux, SunOS, Solaris, Macintosh OS X, VxWorks, or other compatible operating systems. In other embodiments, the computing devices may be controlled by a proprietary operating system. Conventional operating systems control and schedule computer processes for execution, perform memory management, provide file system, networking, I/O services, and provide a user interface functionality, such as a graphical user interface ("GUI"), among other things. The template system 100 may be

hosted and/or executed on one or more computing devices with one or more hardware processors and with any of the previously mentioned operating system software.

Figure 5 is a block diagram that illustrates example components of the template system 100. While Figure 5 refers to the template system 100, any of the other computing devices discussed herein may have some or all of the same or similar components.

The template system 100 may execute software, e.g., standalone software applications, applications within browsers, network applications, etc., whether by the particular application, the operating system, or otherwise. Any of the systems discussed herein may be performed by the template system 100 and/or a similar computing system having some or all of the components discussed with reference to Figure 5.

The template system 100 includes a bus 502 or other communication mechanism for communicating information, and a hardware processor, or multiple processors, 504 coupled with bus 502 for processing information. Hardware processor(s) 504 may be, for example, one or more general purpose microprocessors.

The template system 100 also includes a main memory 506, such as a random access memory (RAM), cache and/or other dynamic storage devices, coupled to bus 502 for storing information and instructions to be executed by processor(s) 504. Main memory 506 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor(s) 504. Such instructions, when stored in storage media accessible to processor(s) 504, render the template system 100 into a special-purpose machine that is customized to perform the operations specified in the instructions. Such instructions, as executed by hardware processors, may implement the methods and systems described herein for sharing security information.

The template system 100 further includes a read only memory (ROM) 508 or other static storage device coupled to bus 502 for storing static information and instructions for processor(s) 504. A storage device 510, such as a magnetic disk, optical disk, or USB thumb drive (Flash drive), etc., is provided and coupled to bus 502 for storing information and instructions. The template storage device 106 and/or the object storage device 130 of Figure 1 may be stored on the main memory 506 and/or the storage device 510.

In some embodiments, the template storage device 106 and/or the object storage device 130 of Figure 1 is a file system, relational database such as, but not limited to, MySql, Oracle,

Sybase, or DB2, and/or a distributed in memory caching system such as, but not limited to, Memcache, Memcached, or Java Caching System.

The template system 100 may be coupled via bus 502 to a display 512, such as a cathode ray tube (CRT) or LCD display or touch screen, for displaying information to a computer user. An input device 514 is coupled to bus 502 for communicating information and command selections to processor 504. One type of input device 514 is a keyboard including alphanumeric and other keys. Another type of input device 514 is a touch screen. Another type of user input device is cursor control 516, such as a mouse, a trackball, a touch screen, or cursor direction keys for communicating direction information and command selections to processor 504 and for controlling cursor movement on display 512. This input device may have two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane. In some embodiments, the same direction information and command selections as cursor control may be implemented via receiving touches on a touch screen without a cursor.

The template system 100 may include a user interface unit to implement a GUI, for example, Figures 4A, 4B, and/or 4C, which may be stored in a mass storage device as executable software codes that are executed by the computing device(s). This and other units may include, by way of example, components, such as software components, object-oriented software components, class components and task components, processes, functions, attributes, procedures, subroutines, segments of program code, drivers, firmware, microcode, circuitry, data, databases, data structures, tables, arrays, and variables.

In general, the word "instructions," as used herein, refers to logic embodied in hardware or firmware, or to a collection of software units, possibly having entry and exit points, written in a programming language, such as, but not limited to, Java, Lua, C, C++, or C#. A software unit may be compiled and linked into an executable program, installed in a dynamic link library, or may be written in an interpreted programming language such as, but not limited to, BASIC, Perl, or Python. It will be appreciated that software units may be callable from other units or from themselves, and/or may be invoked in response to detected events or interrupts. Software units configured for execution on computing devices by their hardware processor(s) may be provided on a computer readable medium, such as a compact disc, digital video disc, flash drive, magnetic disc, or any other tangible medium, or as a digital download (and may be originally stored in a compressed or installable format that requires installation, decompression or decryption prior to execution). Such software code may be stored, partially or fully, on a memory device of the executing computing device, for

execution by the computing device. Software instructions may be embedded in firmware, such as an EPROM. It will be further appreciated that hardware modules may be comprised of connected logic units, such as gates and flip-flops, and/or may be comprised of programmable units, such as programmable gate arrays or processors. Generally, the instructions described herein refer to logical modules that may be combined with other modules or divided into sub-modules despite their physical organization or storage.

The template system 100, or components of it, such as selection unit 108 and/or template processor 109 of Figure 1, may be programmed, via executable code instructions, in a programming language.

The term "non-transitory media," and similar terms, as used herein refers to any media that store data and/or instructions that cause a machine to operate in a specific fashion. Such non-transitory media may comprise non-volatile media and/or volatile media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 510. Volatile media includes dynamic memory, such as main memory 506. Common forms of non-transitory media include, for example, a floppy disk, a flexible disk, hard disk, solid state drive, magnetic tape, or any other magnetic data storage medium, a CD-ROM, any other optical data storage medium, any physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, NVRAM, any other memory chip or cartridge, and networked versions of the same.

Non-transitory media is distinct from but may be used in conjunction with transmission media. Transmission media participates in transferring information between nontransitory media. For example, transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 502. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Various forms of media may be involved in carrying one or more sequences of one or more instructions to processor(s) 504 for execution. For example, the instructions may initially be carried on a magnetic disk or solid state drive of a remote computer. The remote computer may load the instructions into its dynamic memory and send the instructions over a telephone or cable line using a modem. A modem local to the template system 100 may receive the data on the telephone or cable line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and appropriate circuitry can place the data on bus 502. Bus 502 carries the data to

main memory 506, from which the processor(s) 504 retrieves and executes the instructions. The instructions received by main memory 506 may retrieve and execute the instructions. The instructions received by main memory 506 may optionally be stored on storage device 510 either before or after execution by processor(s) 504.

The template system 100 also includes a communication interface 518 coupled to bus 502. Communication interface 518 provides a two-way data communication coupling to a network link 520 that is connected to a local network 522. For example, communication interface 518 may be an integrated services digital network (ISDN) card, cable modem, satellite modem, or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 518 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN (or WAN component to be communicated with a WAN). Wireless links may also be implemented. In any such implementation, communication interface 518 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 520 typically provides data communication through one or more networks to other data devices. For example, network link 520 may provide a connection through local network 522 to a host computer 524 or to data equipment operated by an Internet Service Provider (ISP) 526. ISP 526 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 528. Local network 522 and Internet 528 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 520 and through communication interface 518, which carry the digital data to and from the template system 100, are example forms of transmission media.

A network, such as the network 160 of Figure 1, may comprise, but is not limited to, one or more local area networks, wide area network, wireless local area network, wireless wide area network, the Internet, or any combination thereof.

The template system 100 can send messages and receive data, including program code, through the network(s), network link 520 and communication interface 518. In the Internet example, a server 530 might transmit a requested code for an application program through Internet 528, ISP 526, local network 522 and communication interface 518.

The received code may be executed by processor(s) 504 as it is received, and/or stored in storage device 510, or other non-volatile storage for later execution.

Embodiments of the present disclosure have been described that relate to man-machine interfaces for generating documents and, more specifically, for man-machine interfaces for receiving user selections of data objects and templates to more efficiently generate documents. In some embodiments, the man-machine interfaces described herein include fewer steps, consume less processing and/or memory resources, permit users to have less knowledge of programming languages and/or software development techniques, allow less users or developers to create documents, and/or are more efficient than traditional programmatic approaches to generating reports and/or their corresponding man-machine interfaces. The man machine-interfaces described herein may advantageously reduce the steps and/or cognitive burden of a user because reports and/or documents may be dynamically created according to certain specifications with the use of familiar document editors for mocking up templates and/or through graphical user interfaces of selecting data objects and/or templates. Furthermore, the techniques described herein may permit more efficient and/or faster document generation by consuming less processing and/or memory resources than traditional report generation techniques because there are less computational steps to modify a report and/or only executable code instructions may have to be evaluated and/or executed in the template.

Each of the processes, methods, and algorithms described in the preceding sections may be embodied in, and fully or partially automated by, code instructions executed by one or more computer systems or computer processors comprising computer hardware. The processes and algorithms may be implemented partially or wholly in application-specific circuitry.

The various features and processes described above may be used independently of one another, or may be combined in various ways. All possible combinations and subcombinations are intended to fall within the scope of this disclosure. In addition, certain method or process blocks may be omitted in some implementations. The methods and processes described herein are also not limited to any particular sequence, and the blocks or states relating thereto can be performed in other sequences that are appropriate. For example, described blocks or states may be performed in an order other than that specifically disclosed, or multiple blocks or states may be combined in a single block or state. The example blocks or states may be performed in serial, in parallel, or in some other manner. Blocks or states may be added to or removed from the disclosed example embodiments. The example systems and components described herein may be configured differently than described. For example, elements may be added to, removed from, or rearranged compared to the disclosed example embodiments.

Conditional language, such as, among others, "can," "could," "might," or "may," unless specifically stated otherwise, or otherwise understood within the context as used, is generally intended to convey that certain embodiments include, while other embodiments do not include, certain features, elements and/or steps. Thus, such conditional language is not generally intended to imply that features, elements and/or steps are in any way required for one or more embodiments or that one or more embodiments necessarily include logic for deciding, with or without user input or prompting, whether these features, elements and/or steps are included or are to be performed in any particular embodiment.

Any process descriptions, elements, or blocks in the flow diagrams described herein and/or depicted in the attached figures should be understood as potentially representing units, segments, or portions of code which include one or more executable instructions for implementing specific logical functions or steps in the process. Alternate implementations are included within the scope of the embodiments described herein in which elements or functions may be deleted, executed out of order from that shown or discussed, including substantially concurrently or in reverse order, depending on the functionality involved, as would be understood by those skilled in the art.

It should be emphasized that many variations and modifications may be made to the above-described embodiments, the elements of which are to be understood as being among other acceptable examples. All such modifications and variations are intended to be included herein within the scope of this disclosure. The foregoing description details certain embodiments of the invention. It will be appreciated, however, that no matter how detailed the foregoing appears in text, the invention can be practiced in many ways. As is also stated above, it should be noted that the use of particular terminology when describing certain features or aspects of the invention should not be taken to imply that the terminology is being re-defined herein to be restricted to including any specific characteristics of the features or aspects of the invention with which that terminology is associated. The scope of the invention should therefore be construed in accordance with the appended claims and any equivalents thereof.

CONCLUSIES

1. Een computersysteem voor het genereren van template-based custom documenten, waarbij het computersysteem het volgende omvat:

een inrichting voor opslag van een template; en

één of meer hardware processoren die via uit te voeren code-instructies geprogrammeerd zijn om het volgende te implementeren:

een templategenerator die geconfigureerd is om:

toegang te krijgen tot een placeholder template die één of meer placeholders omvat welke locaties aangeven voor het invoeren van uit te voeren instructies;

uit te voeren instructies te ontvangen om in de placeholder template onder te brengen; en

in de inrichting voor opslag van een template een template op te slaan met ontvangen één of meer stellen uit te voeren instructies die zijn ingevoerd in corresponderende placeholders van de placeholder template;

een templatekeuze-eenheid die geconfigureerd is voor het ontvangen van een keuze van de template en keuze van één of meer dataobjecten om ondergebracht te worden in een gegenereerd custom document op basis van eigenschappen van één of meer dataobjecten;

een template processor die geconfigureerd is om:

de één of meer in de template ondergebrachte stellen uit te voeren instructies te parsen;

elk stel uit te voeren instructies uit

te voeren, waarbij ten minste enkele stellen uit te voeren instructies instructies hebben om toegang te krijgen tot eigenschappen van de gekozen dataobjecten die in één of meer databronnen zijn opgeslagen;

een uitvoer voor elk stel uit te voeren instructies te genereren; en

het custom document te genereren door stellen uit te voeren instructies in de template te vervangen door de uitvoer die gegenereerd wordt door het uitvoeren van corresponderende stellen uit te voeren instructies, waarbij ten minste een gedeelte van de uitvoer eigenschappen van de gekozen dataobjecten en/of samenvattende data met betrekking tot de eigenschappen van de gekozen dataobjecten omvat.

2. Het systeem volgens conclusie 1, waarbij de template een XML based documentopmaak omvat.

3. Het systeem volgens conclusie 1 of conclusie 2, waarbij het custom document een document voor wetshandhaving omvat.

4. Het systeem volgens één van de voorgaande conclusies, waarbij de placeholder template het volgende omvat:
een placeholder voor een naam;
een placeholder voor een afbeelding;
een placeholder voor contactinformatie; en
een placeholder voor een adres.

5. Het systeem volgens conclusie 4, waarbij de placeholder template verder het volgende omvat:
een placeholder voor een alias;
een placeholder voor geassocieerden; en
een placeholder voor eerdere arrestaties.

6. Het systeem volgens één van de voorgaande conclusies, waarbij de template processor het volgende omvat:

een code interface die geconfigureerd is om code in meervoudige programmeertalen te parsen.

7. Het systeem volgens conclusie 6, waarbij de code interface geconfigureerd is voor Groovy.

8. Het systeem volgens één van de voorgaande conclusies, waarbij de dataobjecten snippetobjecten omvatten, waarbij de snippetobjecten dataeigenschappen voor dagvaardingen omvatten.

9. Het systeem volgens één van de voorgaande conclusies, verder omvattende:
een user interface eenheid die geconfigureerd is om één of meer user interfaces te configureren welke geconfigureerd zijn om één of meer te kiezen dataobjecten te laten zien, waarbij de keuze van de één of meer in het custom document onder te brengen dataobjecten, wordt ontvangen via de één of meer user interfaces.

10. Het systeem volgens conclusie 9, waarbij de één of meer user interfaces zijn geconfigureerd om:
uit te voeren code-instructies van een interactieve commandolijn te ontvangen, waarbij ten minste enkele uit te voeren code-instructies geconfigureerd zijn om het systeem toegang te laten krijgen tot eigenschappen van de gekozen dataobjecten wanneer deze worden uitgevoerd; en
de uitvoer te tonen van de ontvangen uit te voeren code-instructies in antwoord op invoer van de interactieve commandolijn.

11. Het systeem volgens conclusie 2 of een van conclusie 2 afhankelijke conclusie, verder omvattende:
een XML document editor, waarbij de XML document editor geconfigureerd is om invoer van een gebruiker te ontvangen om de XML documenten van een XML based document te wijzigen.

12. Een werkwijze voor het genereren van template based custom documenten, waarbij de werkwijze het volgende omvat:

het ontvangen van een keuze van een template, waarbij de template één of meer stellen uit te voeren instructies omvat;

het ontvangen van een keuze van één of meer dataobjecten om toegang te krijgen in antwoord op het uitvoeren van de één of meer stellen uit te voeren instructies;

het door middel van een computersysteem met één of meer computerprocessoren en een elektronisch opslagapparaat uitvoeren van elk stel uit te voeren instructies, waarbij ten minste enige stellen uit te voeren instructies instructies omvatten om toegang te krijgen tot eigenschappen van ten minste enkele gekozen dataobjecten;

het genereren van een uitvoer voor elk stel uit te voeren instructies; en

het genereren van een custom document door het vervangen van stellen uit te voeren instructies in de template door de corresponderende gegenereerde uitvoer, inclusief eigenschappen van de gekozen dataobjecten en/of samenvattende data met betrekking tot de eigenschappen van de gekozen dataobjecten.

13. De werkwijze volgens conclusie 12, waarbij de template een XML based documentopmaak omvat.

14. De werkwijze volgens conclusie 13, verder omvattende:

het aan een gebruiker van het computersysteem verschaffen van een XML editing user interface, waarbij de XML editing user interface geconfigureerd is om gebruikersinvoer te ontvangen om de XML documenten van een XML based document te wijzigen;

het ontvangen van invoer van de gebruiker van statische tekst

in een placeholder template, waarbij de statische tekst ondergebracht moet worden in het gegenereerde custom document;
het ontvangen van invoer van een placeholder karakterstring op een bijzondere locatie van een placeholder template;
het verschaffen van een XML view van de placeholder template, waarbij de XML view elementen omvat die bruikbaar zijn door een woordprocessor om een afbeelding van de placeholder template te genereren;
het verschaffen van identifiers om de placeholder karakterstring in de XML view te lokaliseren;
het ontvangen van een eerste stel uit te voeren instructies die de placeholder karakterstring in de XML view vervangen;
het als de template opslaan van de placeholder template met het eerste stel uit te voeren instructies die de placeholder karakterstring vervangen.

15. De werkwijze volgens één van de conclusies 12-14, waarbij de keuze van de één of meer dataobjecten die in het custom document moeten worden ondergebracht, worden ontvangen via één of meer user interfaces.

16. De werkwijze volgens één van de conclusies 12-15, waarbij het custom document een rap sheet omvat met informatie over de criminele geschiedenis van een individu.

17. Niet-tijdelijke computeropslag, omvattende instructies om een computersysteem template-based custom documenten te laten genereren door:
het ontvangen van een template dat gewijzigd werd door placeholders te vervangen door stellen uit te voeren instructies;
het ontvangen van een keuze van de template;
het ontvangen van één of meer datatobjecten om onder te brengen in een gegenereerd custom document op basis van

eigenschappen van de één of meer dataobjecten;
het door middel van het computersysteem parsen van de één of meer stellen uit te voeren instructies van de template;
het uitvoeren van elk stel uit te voeren instructies, waarbij ten minste enkele stellen uit te voeren instructies instructies omvatten om toegang te krijgen tot eigenschappen van de gekozen dataobjecten die van één of meer databronnen zijn opgeslagen;
het genereren van een uitvoer voor elk stel uit te voeren instructies; en
het genereren van een custom document door stellen uit te voeren instructies in de template te vervangen door de corresponderende gegenereerde uitvoer, inclusief eigenschappen van de gekozen dataobjecten en/of samenvattende data met betrekking tot de eigenschappen van de gekozen dataobjecten.

18. Niet-tijdelijke computeropslag volgens conclusie 17, waarbij de template een XML based documentopmaak omvat.

19. Niet-tijdelijke computeropslag volgens conclusie 17 of conclusie 18, waarbij het custom document een document voor wetshandhaving omvat.

20. Niet-tijdelijke computeropslag volgens één van de conclusies 17-19, waarbij de keuze van de één of meer in het custom document onder te brengen dataobjecten wordt ontvangen via één of meer user interfaces.

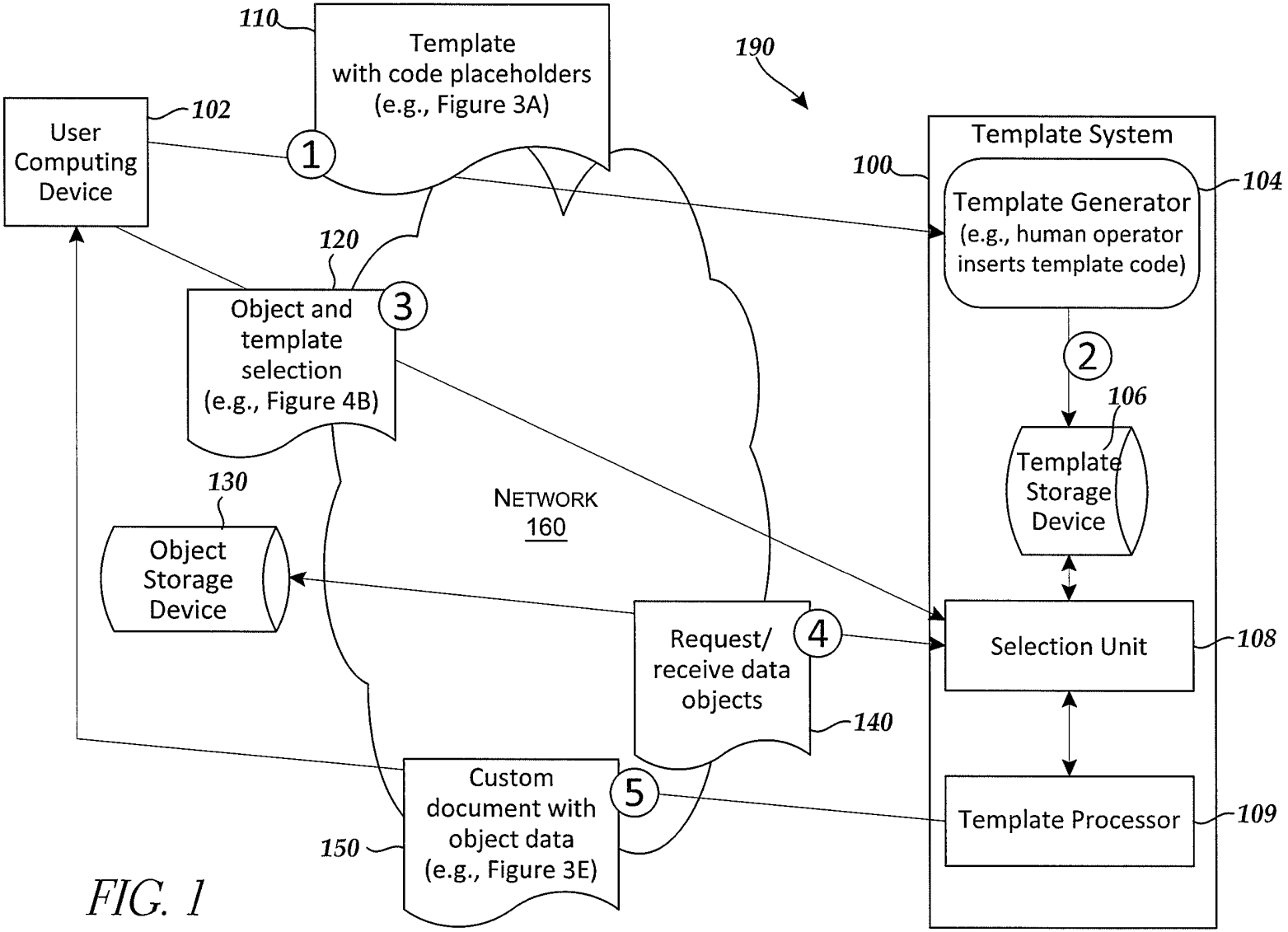
110
Template
with code placeholders
(e.g., Figure 3A)
190
User
Computing
Device
102
1
Template System
100
Template Generator
(e.g., human operator
inserts template code)
104
120
Object and
template
selection
(e.g., Figure 4B)
3
2
106
Template
Storage
Device
130
NETWORK
160
Object
Storage
Device
Request/
receive data
objects
4
Selection Unit
108
140
Custom
document with
object data
(e.g., Figure 3E)
5
150
Template Processor
109

FIG. 1

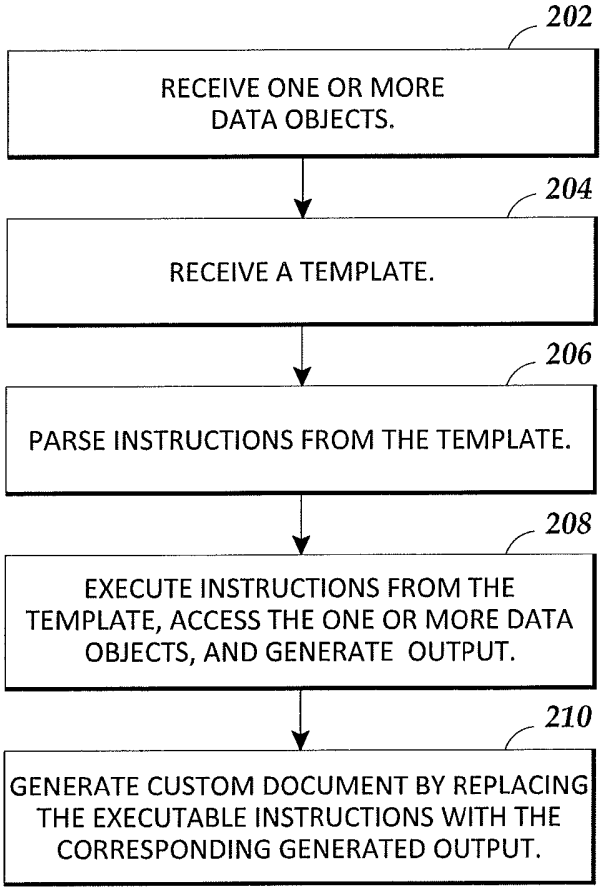
202
RECEIVE ONE OR MORE DATA OBJECTS.
204
RECEIVE A TEMPLATE.
206
PARSE INSTRUCTIONS FROM THE TEMPLATE.
208
EXECUTE INSTRUCTIONS FROM THE TEMPLATE, ACCESS THE ONE OR MORE DATA OBJECTS, AND GENERATE OUTPUT.
210
GENERATE CUSTOM DOCUMENT BY REPLACING THE EXECUTABLE INSTRUCTIONS WITH THE CORRESPONDING GENERATED OUTPUT.

FIG. 2

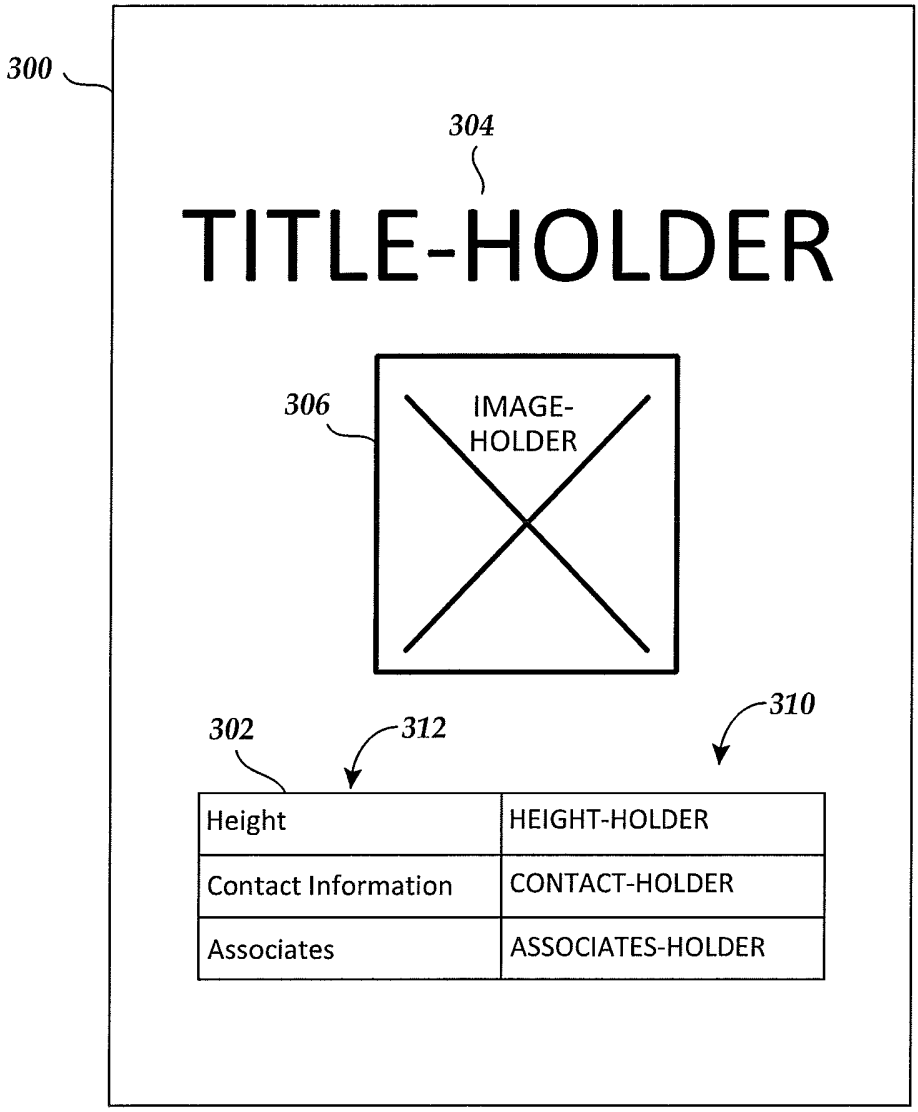
300
304
TITLE-HOLDER
306
IMAGE-HOLDER
302
312
310
Height
HEIGHT-HOLDER
Contact Information
CONTACT-HOLDER
Associates
ASSOCIATES-HOLDER

FIG. 3A

FIG. 3B

310A

```
<document>
        |
        |
    <paragraph>
        <text>NAME-HOLDER</text>
    </paragraph>
    <paragraph>
        <image name="image-HOLDER.png" />
    </paragraph>
    <paragraph>
        <table>
            <row>
                <tableData>Height</tableData>
                <tableData>HEIGHT-HOLDER</tableData>
            </row>
            <row>
                <tableData>Contact Information</tableData>
                <tableData>CONTACT-HOLDER</tableData>
            </row>
            <row>
                <tableData>Associates</tableData>
                <tableData>ASSOCIATES-HOLDER</tableData>
            </row>
        </table>
    </paragraph>
        |
        |
</document>
```

FIG. 3C

310B

```
<document>
        ¦
<% for each object in objects %>

        <paragraph>
                <text><%= print(object.getName) %></text>
        </paragraph>
        <paragraph>
                <image name="<%= print(object.getPicture) %>" />
        </paragraph>
        <paragraph>
                <table>
                        <row>
316                                     <tableData>Height</tableData>
                                        <tableData>
<%
records = object.getRecords;
height = frequencyMap(records, "height").getMostCommon;
print(height);
%>
                                        </tableData>
                                </row>
                                <row>
312                                     <tableData>Contact Information</tableData>
                                        <tableData>
<%
if object.getPhoneNumber is not null
314 println("Phone Number: " + object.getPhoneNumber);

if object.getEmail is not null
        println("Email: " + object.getEmail);
%>
                                        </tableData>
                                </row>
                                <row>
                                        <tableData>Associates</tableData>
                                        <tableData><%= print(object.getAssociates.join
                                         (",")) %></tableData>
                                </row>
                        </table>
        </paragraph>
<% end %>
        ¦
</document>
```

FIG. 3D

320

```
<document>
322
        <paragraph>
                <text>John Doe</text>
        </paragraph>
        <paragraph>
                <image name="johnDoe.png" />
        </paragraph>
        <paragraph>
                <table>
                        <row>
                                <tableData>Height</tableData>
                                <tableData>5'7"</tableData>
                        </row>
                        <row>
                                <tableData>Contact Information</tableData>
                                <tableData>Phone Number: (123) 111-1111</tableData>
                        </row>
                        <row>
                                <tableData>Associates</tableData>
                                <tableData>Jane Doe, Bob Doe, Jerry Dole</tableData>
                        </row>
                        </table>
        </paragraph>
</document>
```

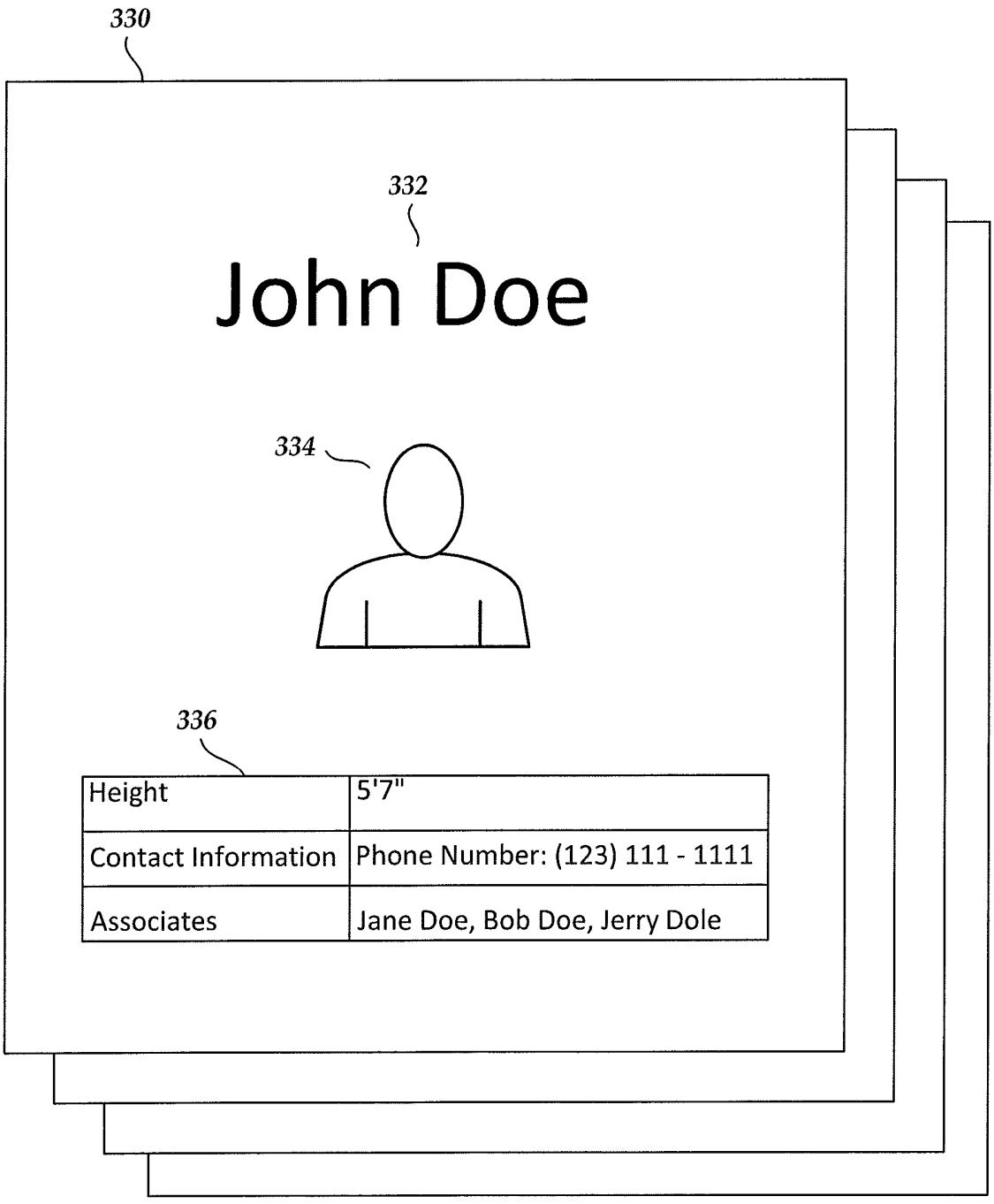
330
332
John Doe
334
336
Height
5'7"
Contact Information
Phone Number: (123) 111 - 1111
Associates
Jane Doe, Bob Doe, Jerry Dole

FIG. 3E

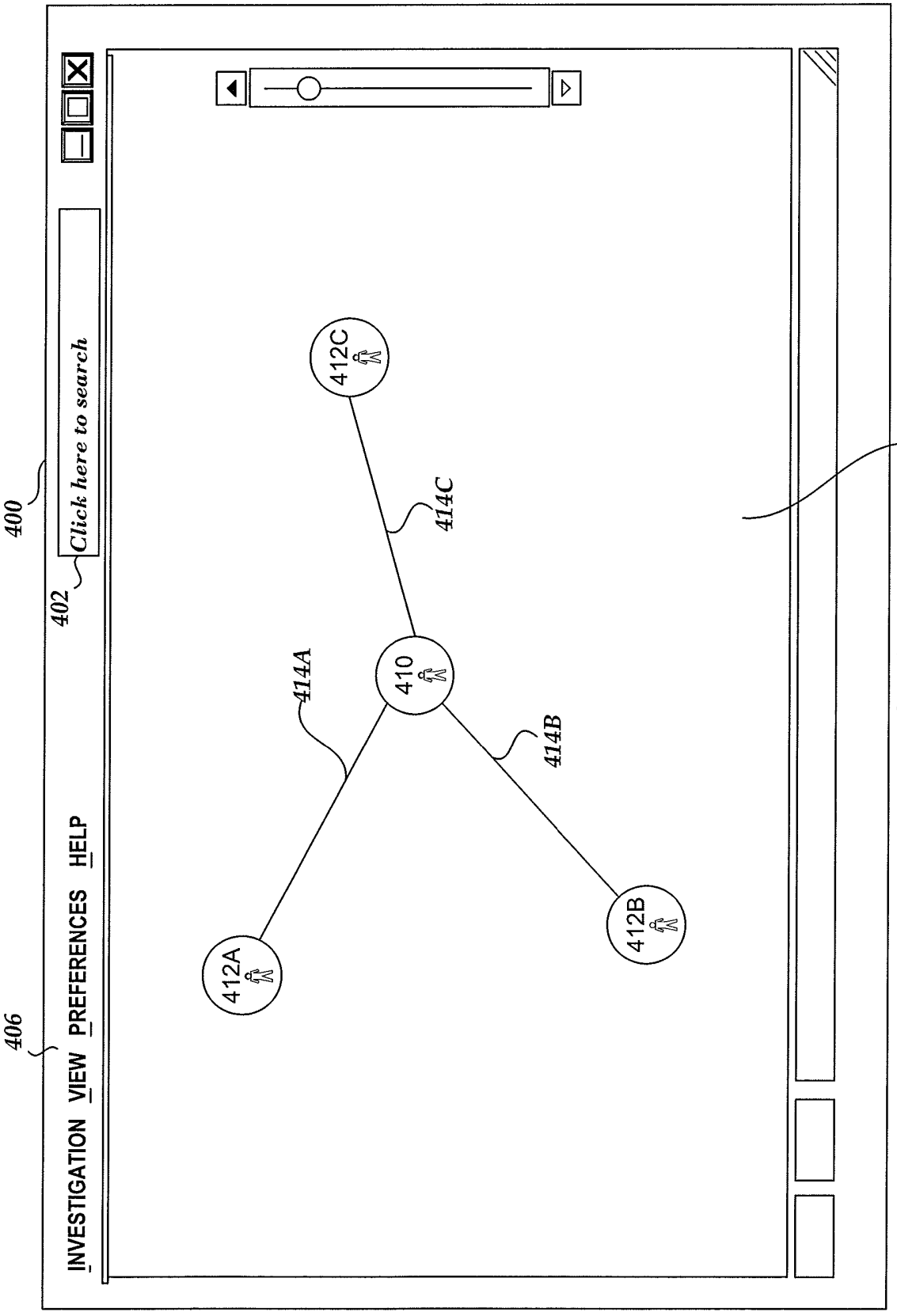
400
406
INVESTIGATION VIEW PREFERENCES HELP
402
Click here to search
404
412A
410
412C
412B
414A
414B
414C

FIG. 4A

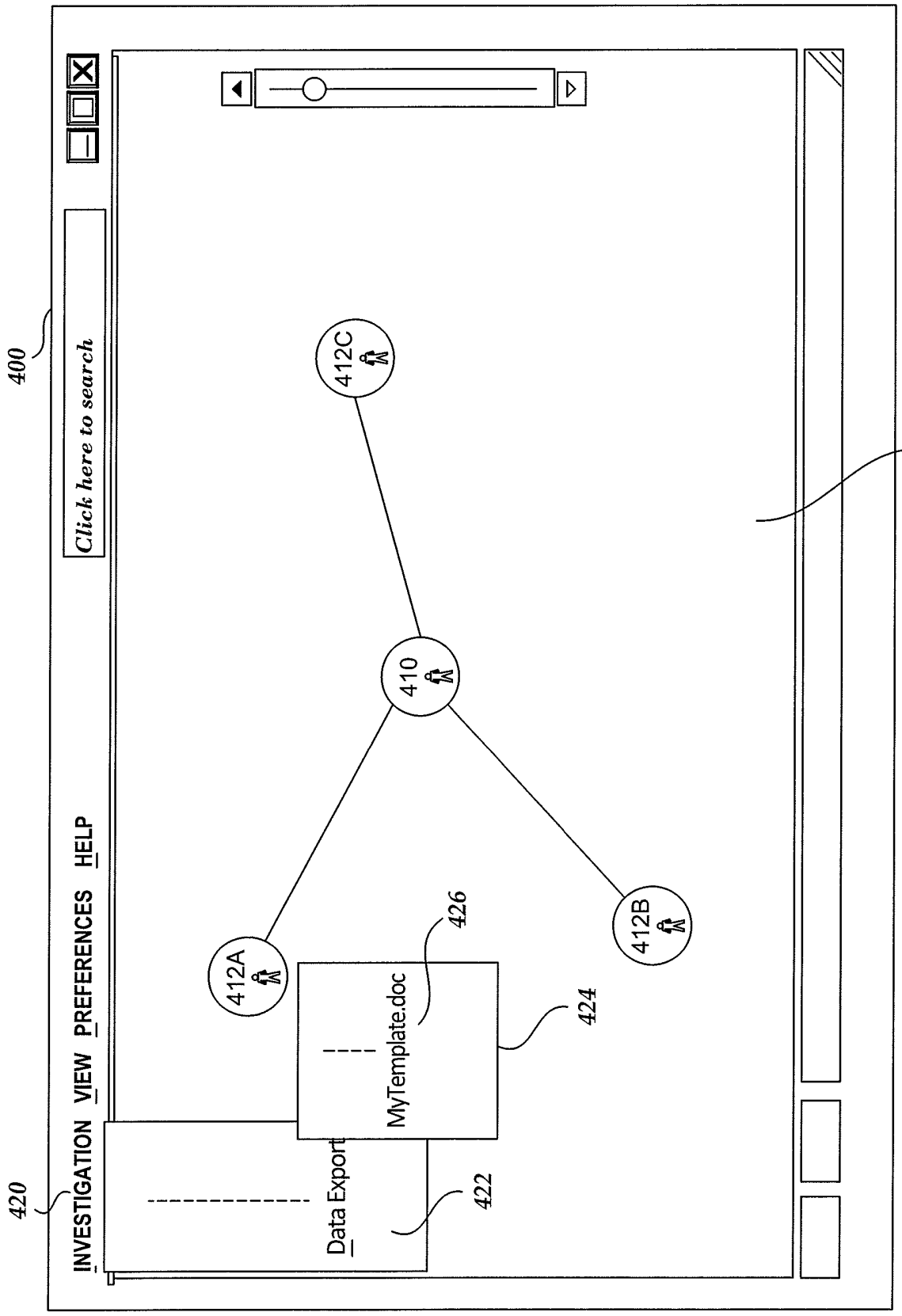
400
INVESTIGATION VIEW PREFERENCES HELP
Click here to search
420
Data Export
422
MyTemplate.doc
426
424
412A
410
412C
412B
404

FIG. 4B

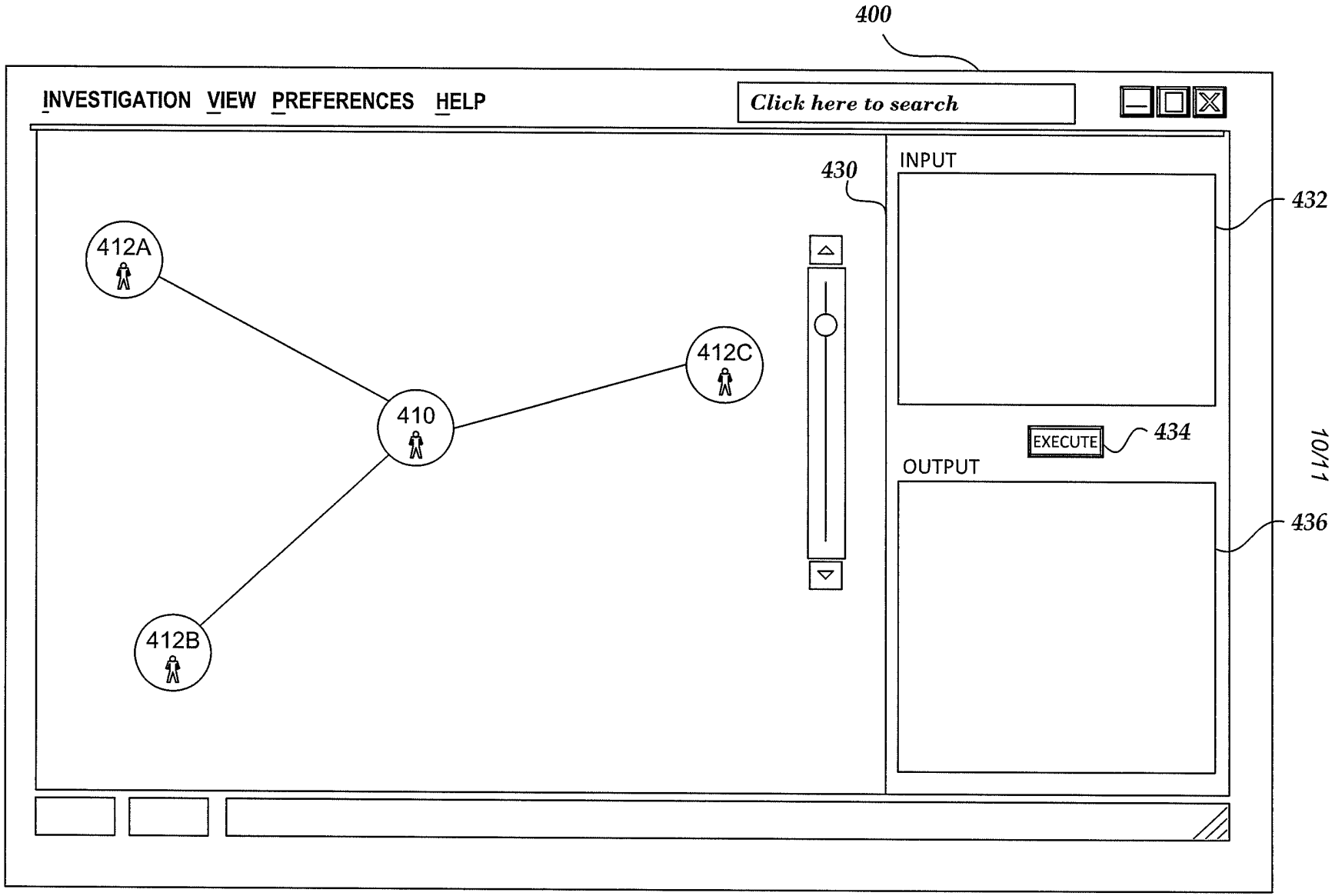
400
INVESTIGATION VIEW PREFERENCES HELP
Click here to search
430
INPUT
432
412A
412C
410
412B
EXECUTE
434
OUTPUT
436
10/11

FIG. 4C

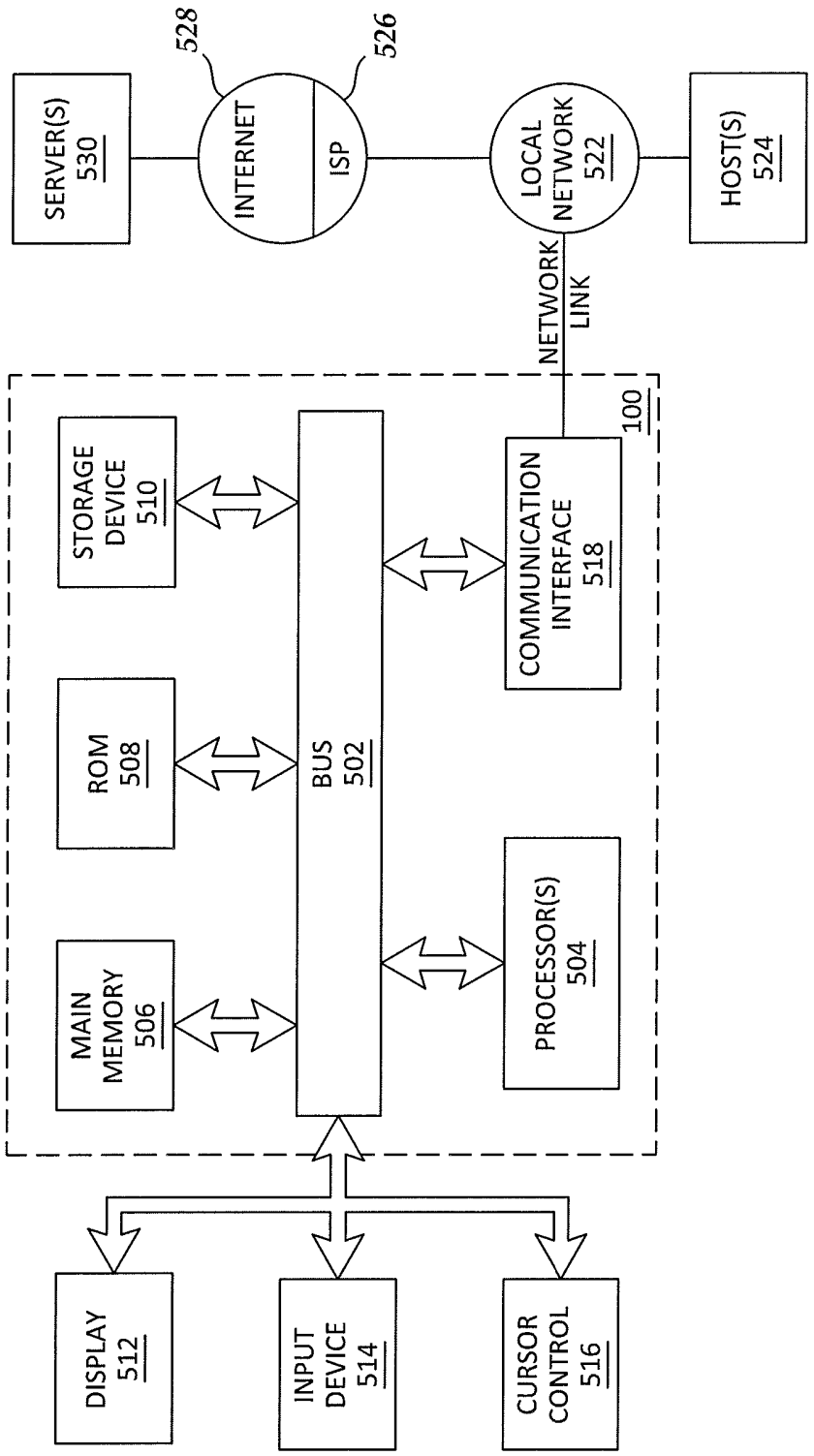
SERVER(S)
530
528
INTERNET
ISP
526
LOCAL
NETWORK
522
HOST(S)
524
NETWORK
LINK
100
STORAGE
DEVICE
510
ROM
508
MAIN
MEMORY
506
BUS
502
COMMUNICATION
INTERFACE
518
PROCESSOR(S)
504
DISPLAY
512
INPUT
DEVICE
514
CURSOR
CONTROL
516

FIG. 5