US 20080263531A1

(19) **United States**

(12) **Patent Application Publication**

Perry et al.

(10) **Pub. No.: US 2008/0263531 A1**

(43) **Pub. Date: Oct. 23, 2008**

(54) AUTOMATIC RUNTIME CONTROL BINDING

(75) Inventors: **Carl Yates Perry**, Woodinville, WA (US); **Jeffrey Michael Derstadt**, Seattle, WA (US); **Andrew J. Conrad**, Sammamish, WA (US); **Amanda K. Silver**, Seattle, WA (US); **Paul A. Vick**, Seattle, WA (US); **Shyamalan Pather**, Seattle, WA (US); **David E. Sceppa**, Seattle, WA (US)

Correspondence Address:
AMIN. TUROCY & CALVIN, LLP
24TH FLOOR, NATIONAL CITY CENTER, 1900 EAST NINTH STREET
CLEVELAND, OH 44114 (US)

(73) Assignee: **MICROSOFT CORPORATION**, Redmond, WA (US)

(21) Appl. No.: **11/847,140**

(22) Filed: **Aug. 29, 2007**

Related U.S. Application Data

(60) Provisional application No. 60/913,180, filed on Apr. 20, 2007, provisional application No. 60/913,183, filed on Apr. 20, 2007, provisional application No. 60/913,186, filed on Apr. 20, 2007.

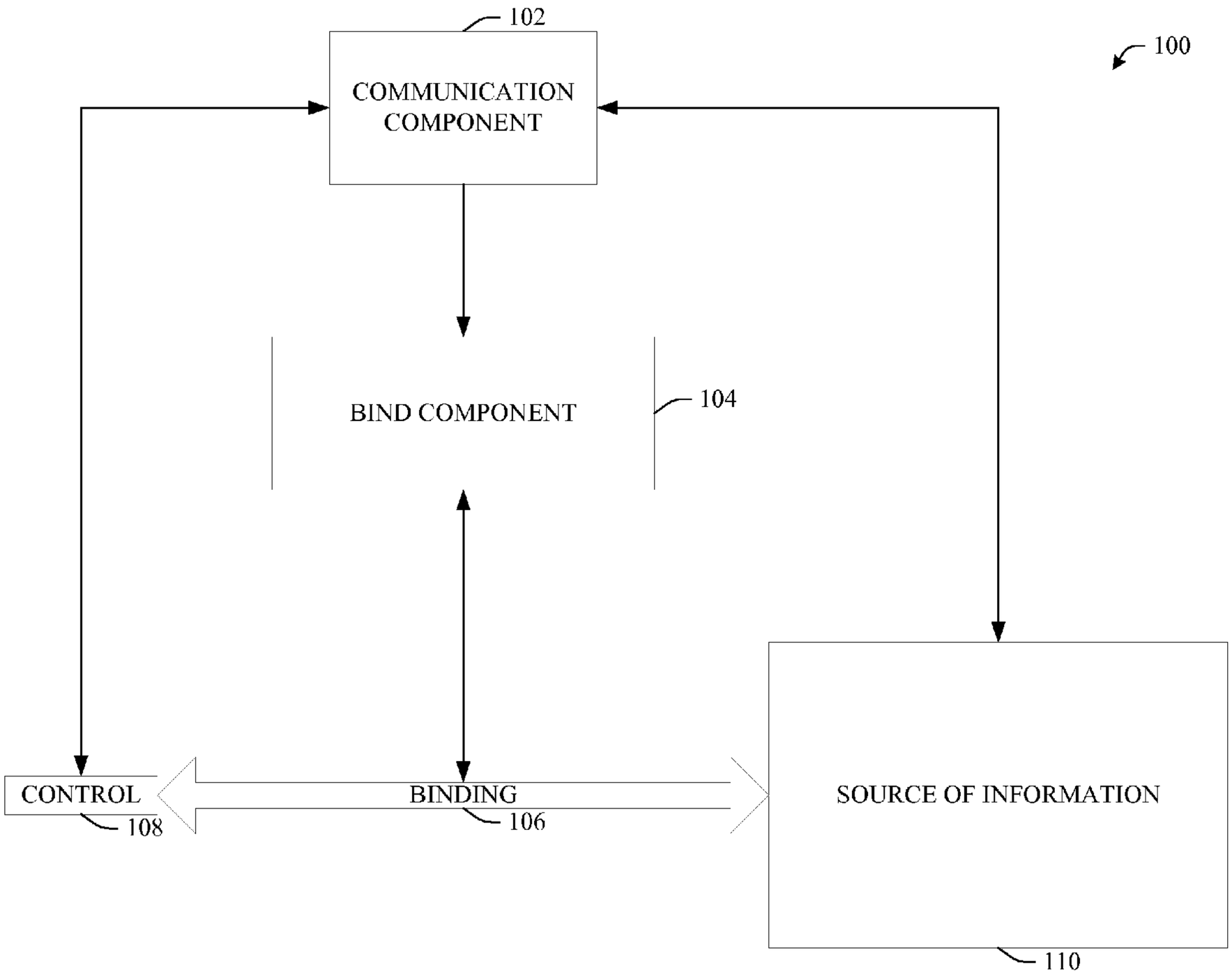
Publication Classification

(51) **Int. Cl.**
G06F 9/44 (2006.01)

(52) **U.S. Cl.** **717/165**

(57) **ABSTRACT**

Binding can automatically take place between controls and information sources at runtime. This minimizes an amount of code that is to be generated by a user and thus can lower errors from the code. In addition, some wizards that allow binding without code writing can become problematic since some information is not available at runtime (e.g., when the wizard operates prior to runtime.) Relevant information is received and at least one binding is created based off received information.



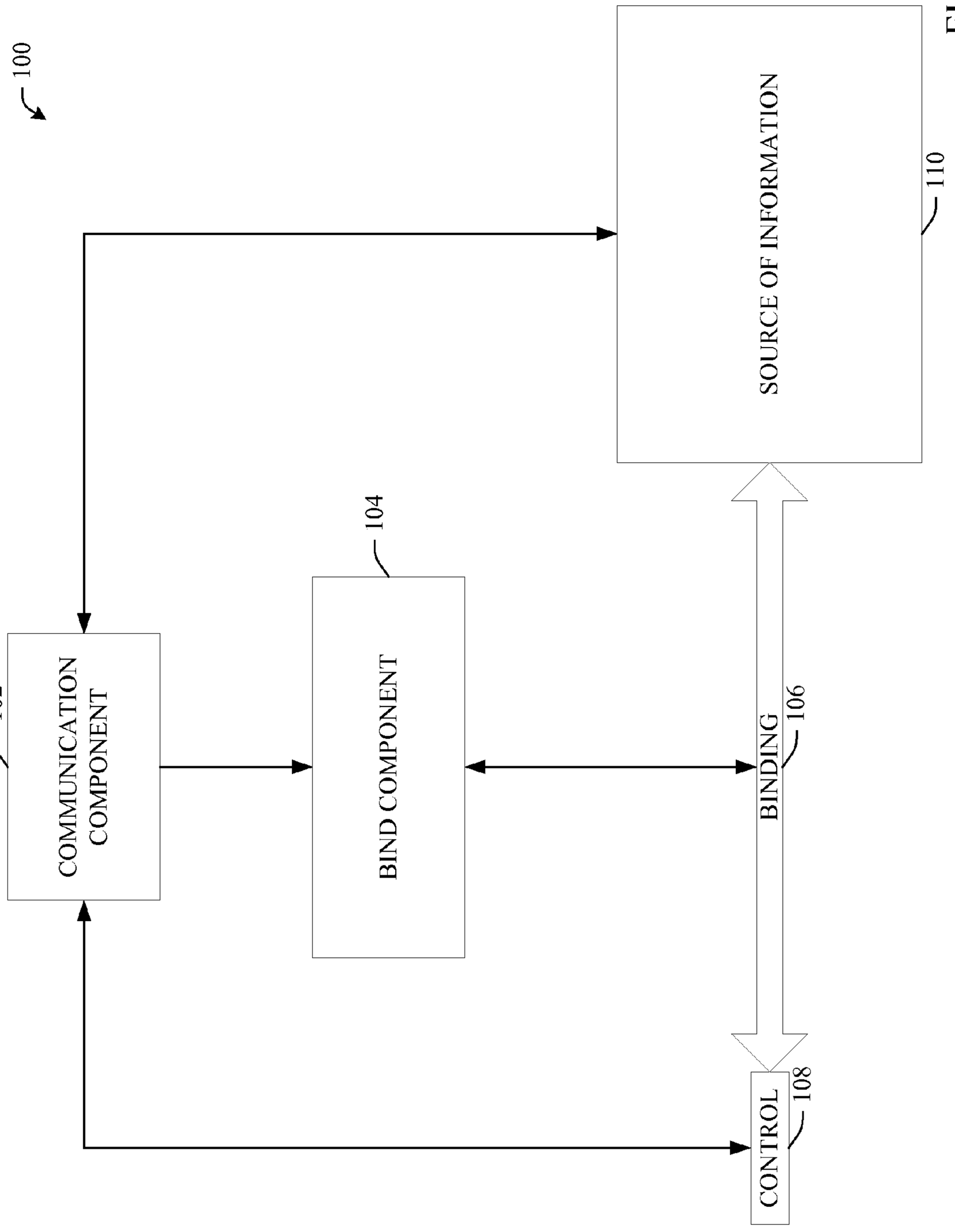

100
SOURCE OF INFORMATION
110
COMMUNICATION COMPONENT
102
BIND COMPONENT
104
BINDING
106
CONTROL
108

FIG. 1

FIG. 2

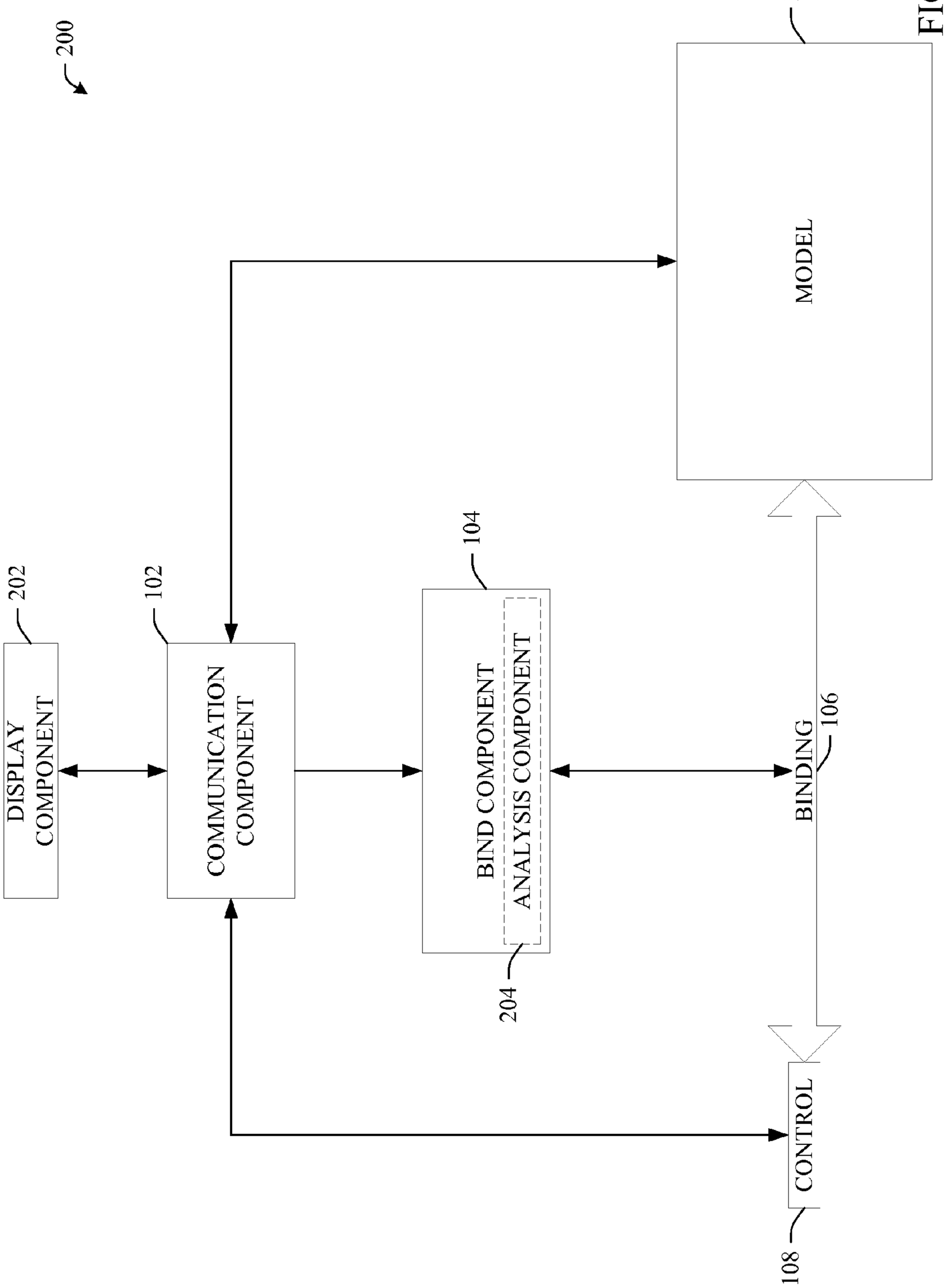

200
206
MODEL
DISPLAY COMPONENT
202
COMMUNICATION COMPONENT
102
BIND COMPONENT
ANALYSIS COMPONENT
104
204
BINDING
106
CONTROL
108

FIG. 3

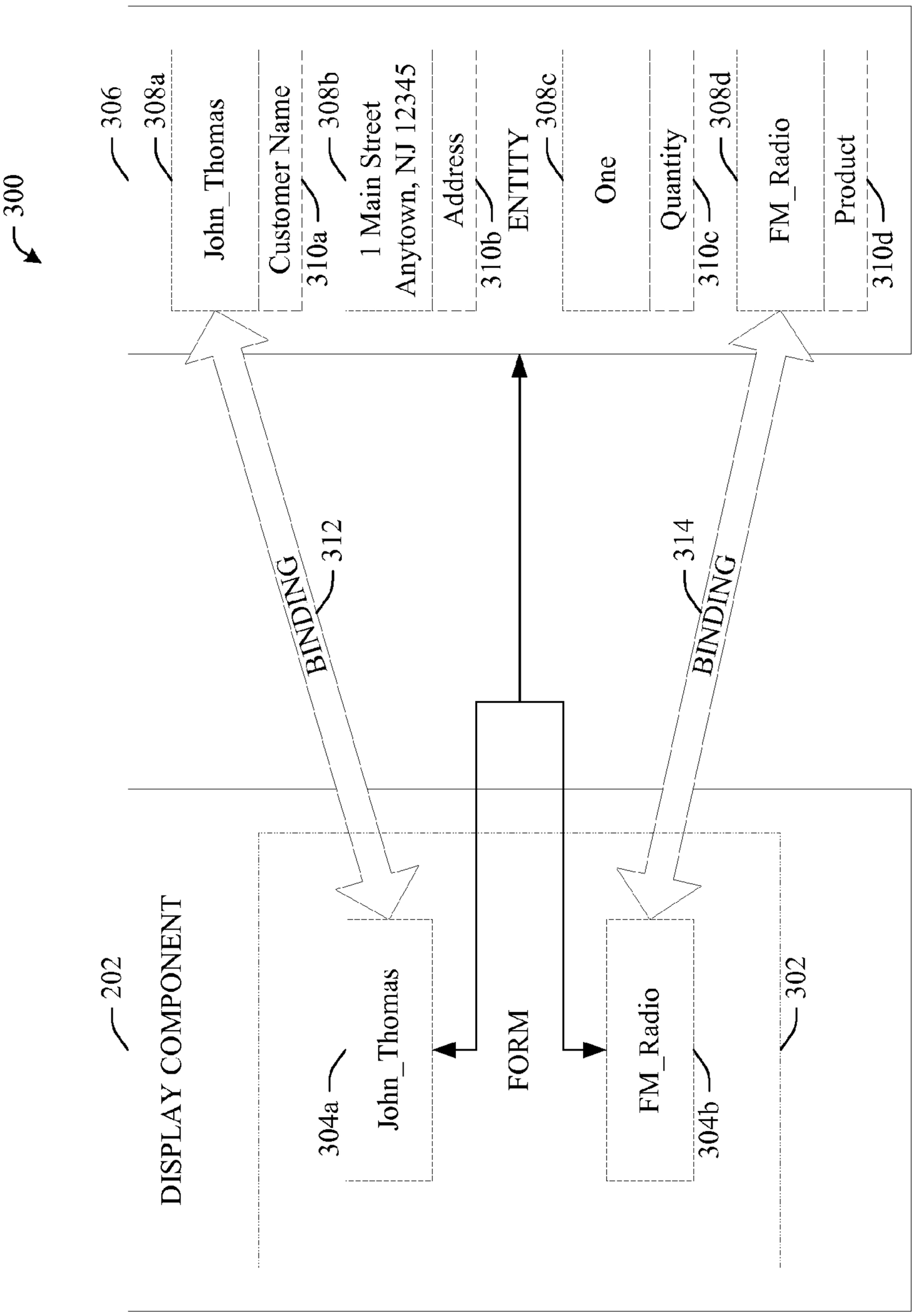
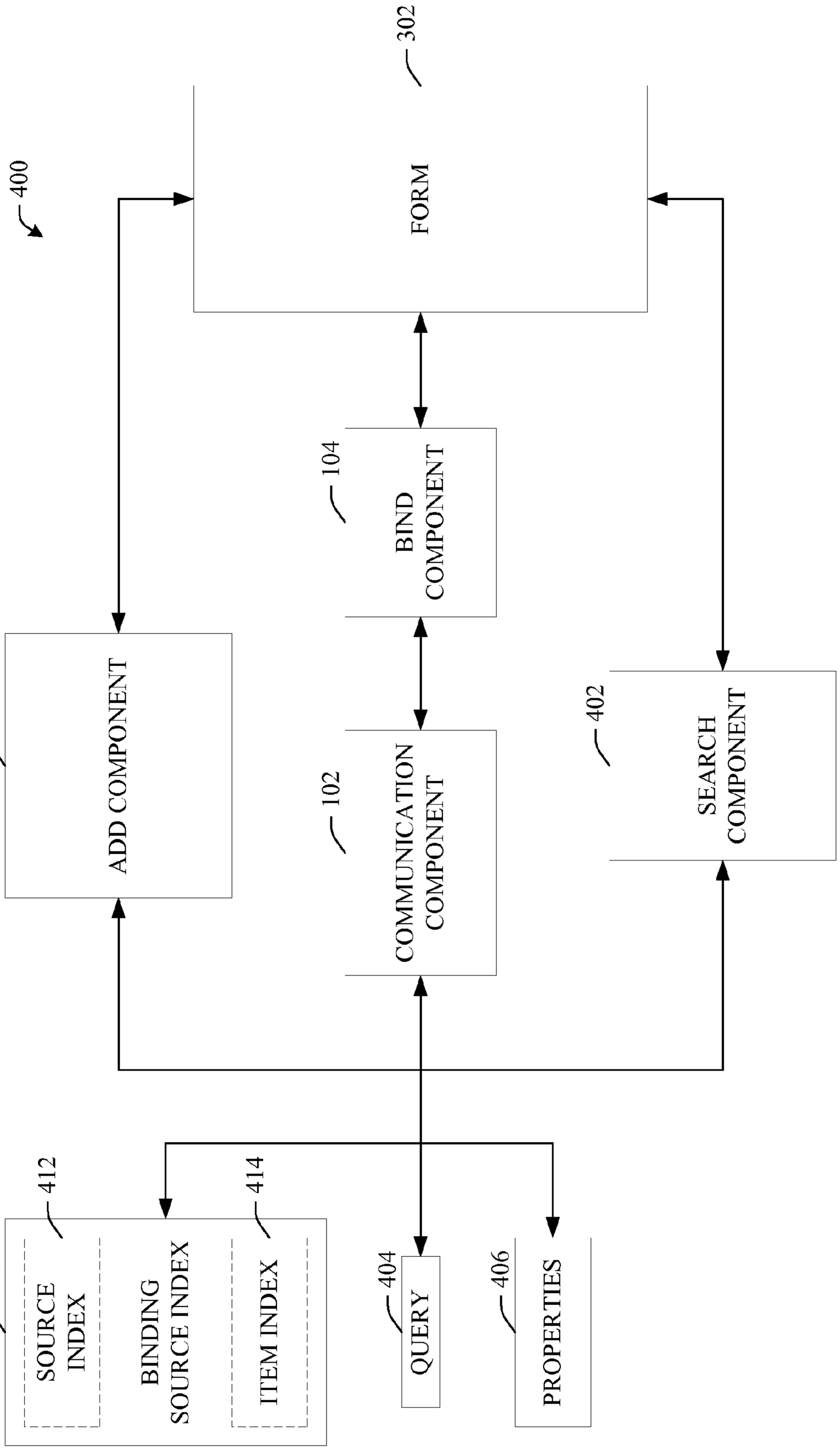

300
306
308a
John_Thomas
Customer Name
310a
308b
1 Main Street
Anytown, NJ 12345
Address
310b
ENTITY
308c
One
Quantity
310c
308d
FM_Radio
Product
310d
BINDING
312
BINDING
314
202
DISPLAY COMPONENT
304a
John_Thomas
FORM
FM_Radio
304b
302

FIG. 4


400
302
FORM
104
BIND COMPONENT
102
COMMUNICATION COMPONENT
402
SEARCH COMPONENT
408
ADD COMPONENT
410
412
SOURCE INDEX
BINDING SOURCE INDEX
414
ITEM INDEX
404
QUERY
406
PROPERTIES

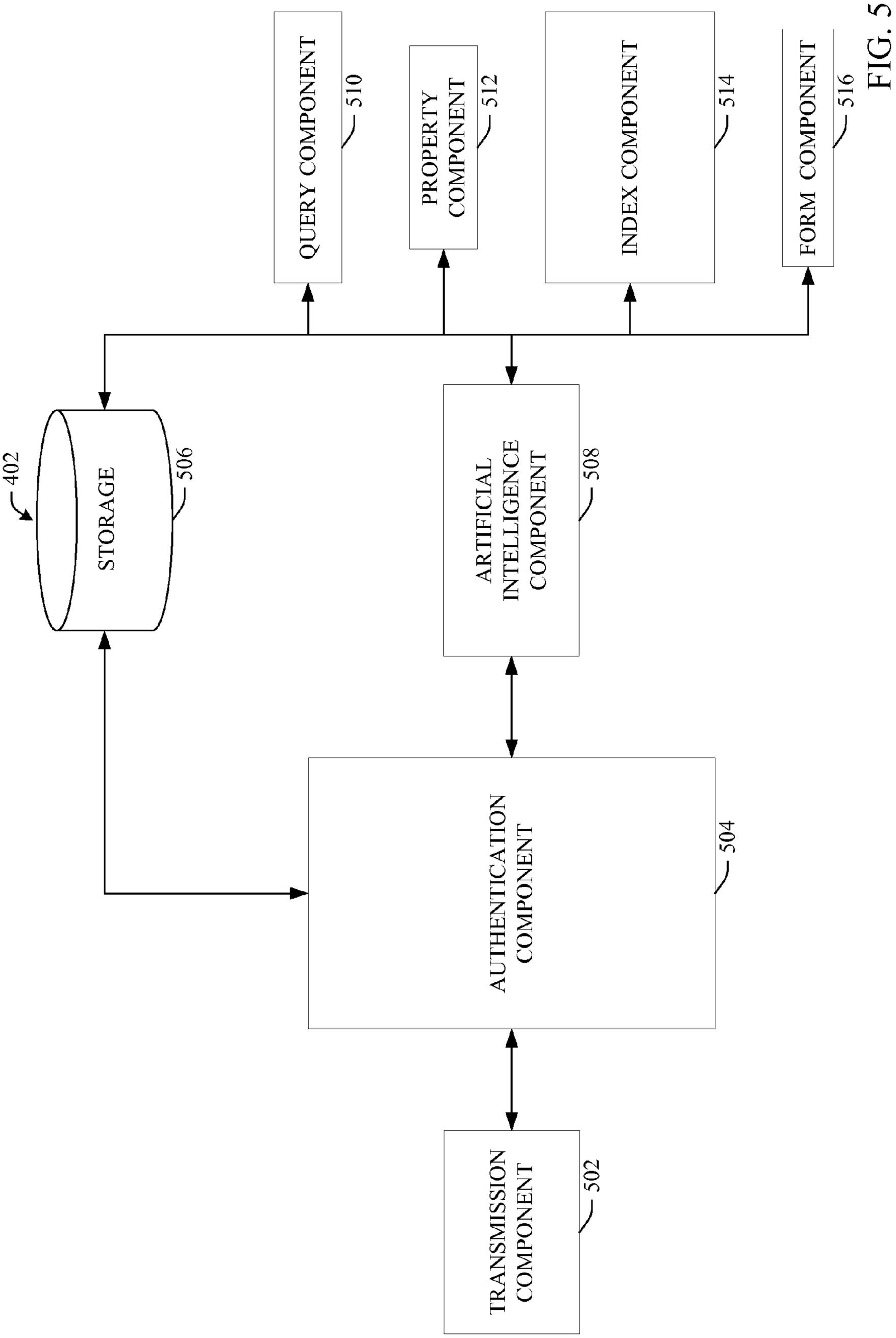

QUERY COMPONENT
510
PROPERTY COMPONENT
512
INDEX COMPONENT
514
FORM COMPONENT
516
402
STORAGE
506
ARTIFICIAL INTELLIGENCE COMPONENT
508
AUTHENTICATION COMPONENT
504
TRANSMISSION COMPONENT
502

FIG. 5

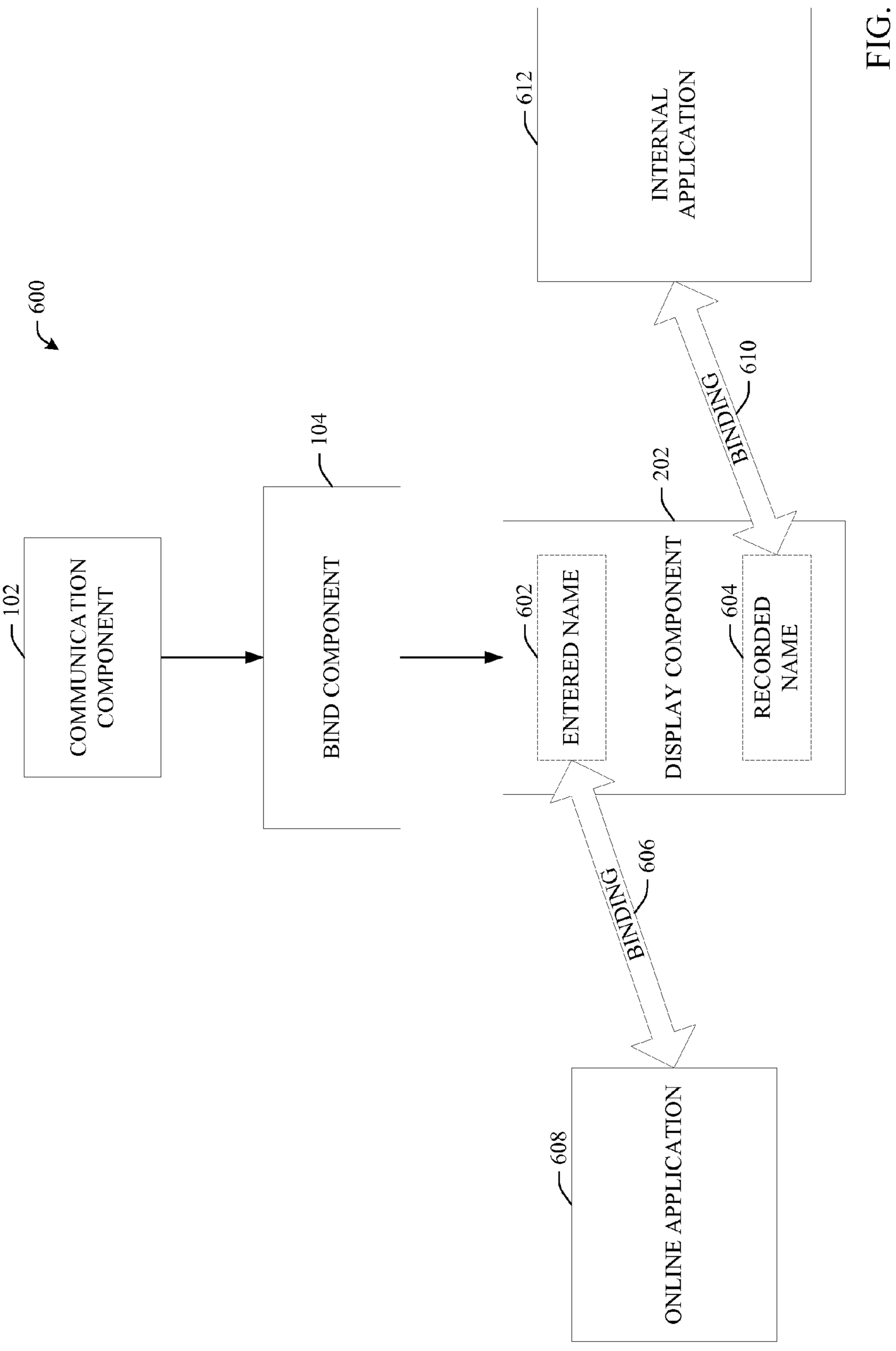

600
102
COMMUNICATION COMPONENT
104
BIND COMPONENT
202
DISPLAY COMPONENT
602
ENTERED NAME
604
RECORDED NAME
606
BINDING
608
ONLINE APPLICATION
610
BINDING
612
INTERNAL APPLICATION

FIG. 6

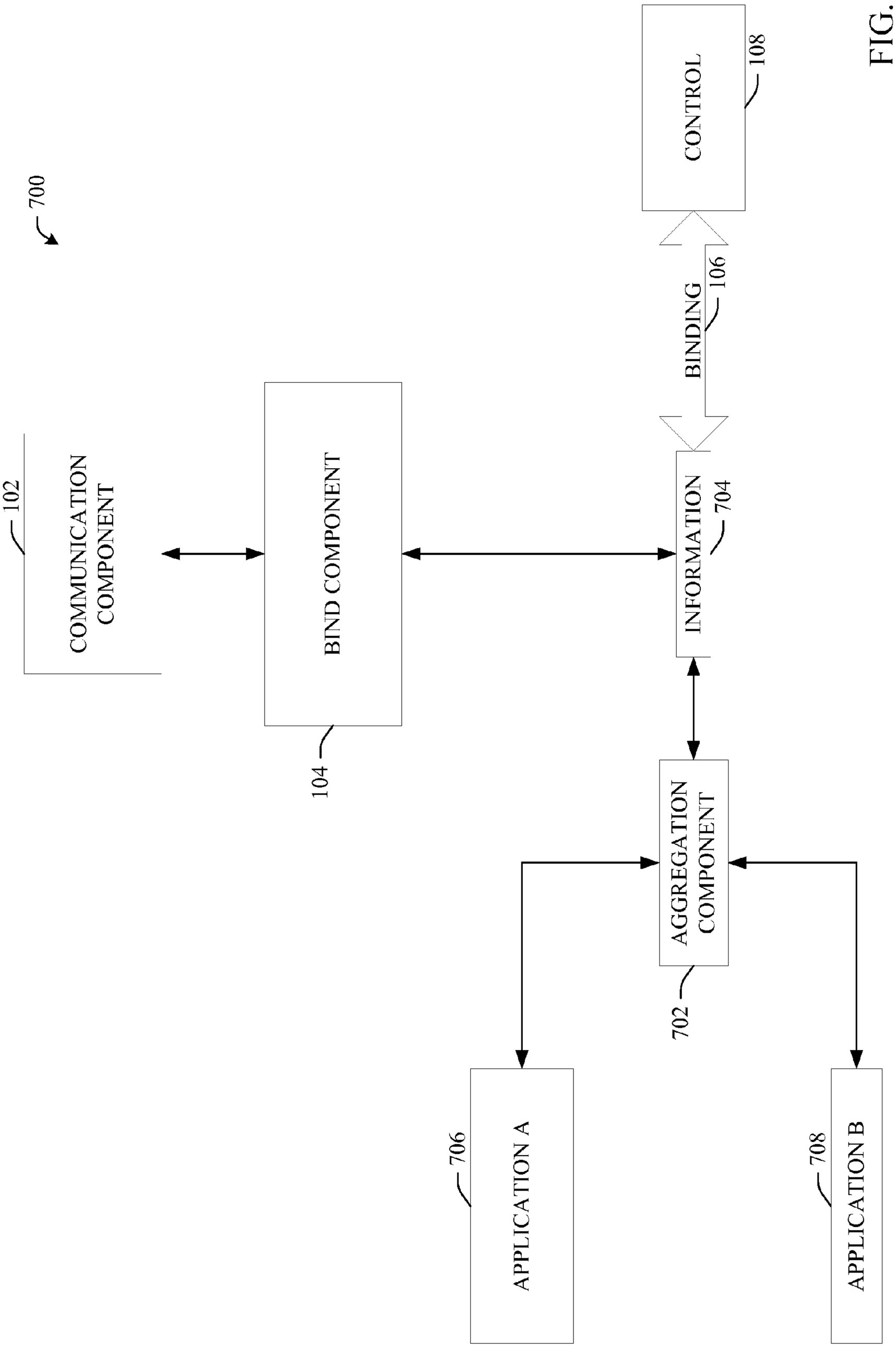
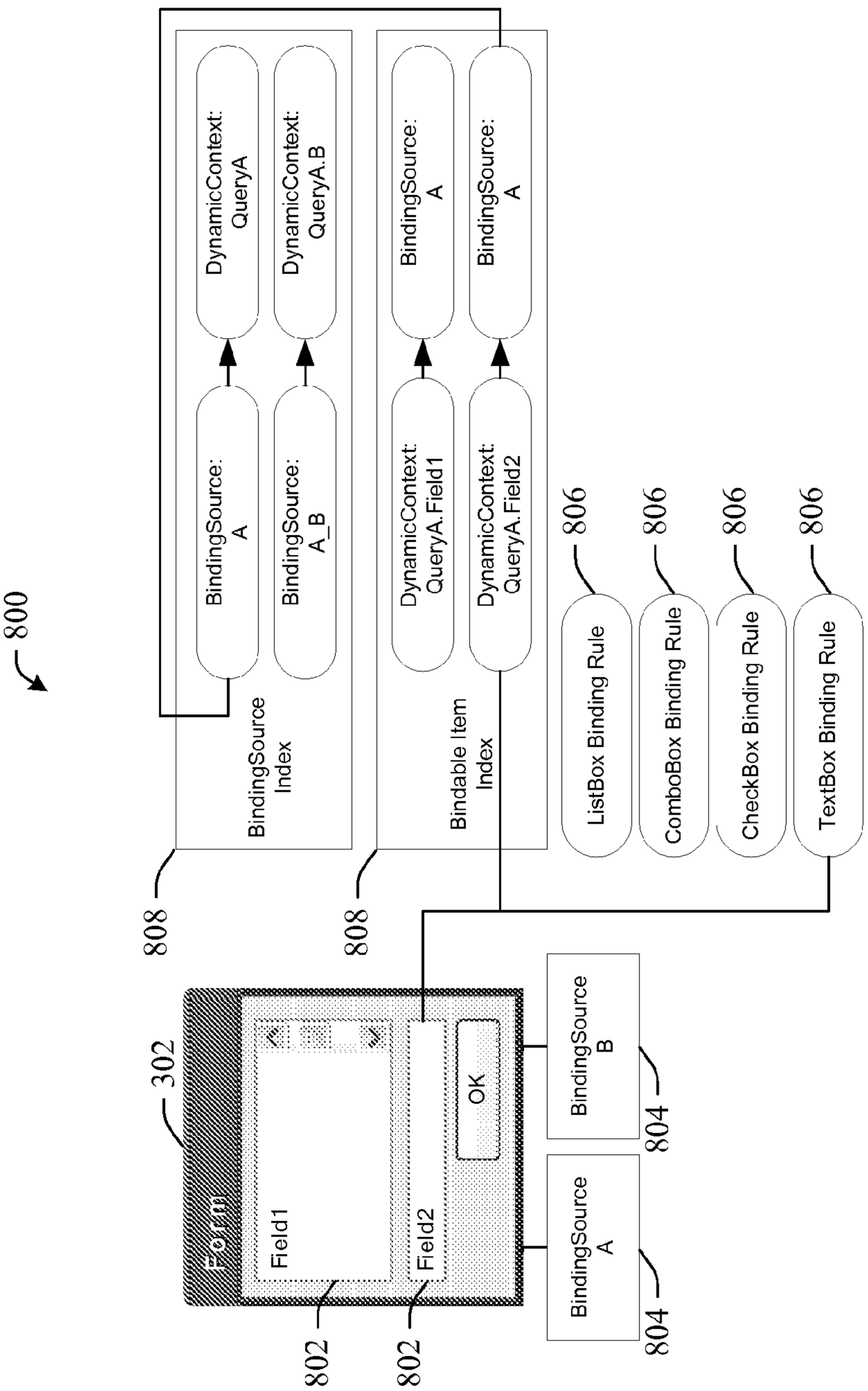

700
102
COMMUNICATION COMPONENT
104
BIND COMPONENT
CONTROL
108
BINDING
106
INFORMATION
704
AGGREGATION COMPONENT
702
APPLICATION A
706
APPLICATION B
708

FIG. 7

FIG. 8


800
BindingSource Index
BindingSource: A
BindingSource: A_B
DynamicContext: QueryA
DynamicContext: QueryA.B
808
Bindable Item Index
DynamicContext: QueryA.Field1
DynamicContext: QueryA.Field2
BindingSource: A
BindingSource: A
808
806
ListBox Binding Rule
ComboBox Binding Rule
CheckBox Binding Rule
TextBox Binding Rule
806
806
806
302
Form
Field1
Field2
OK
802
802
BindingSource A
BindingSource B
804
804

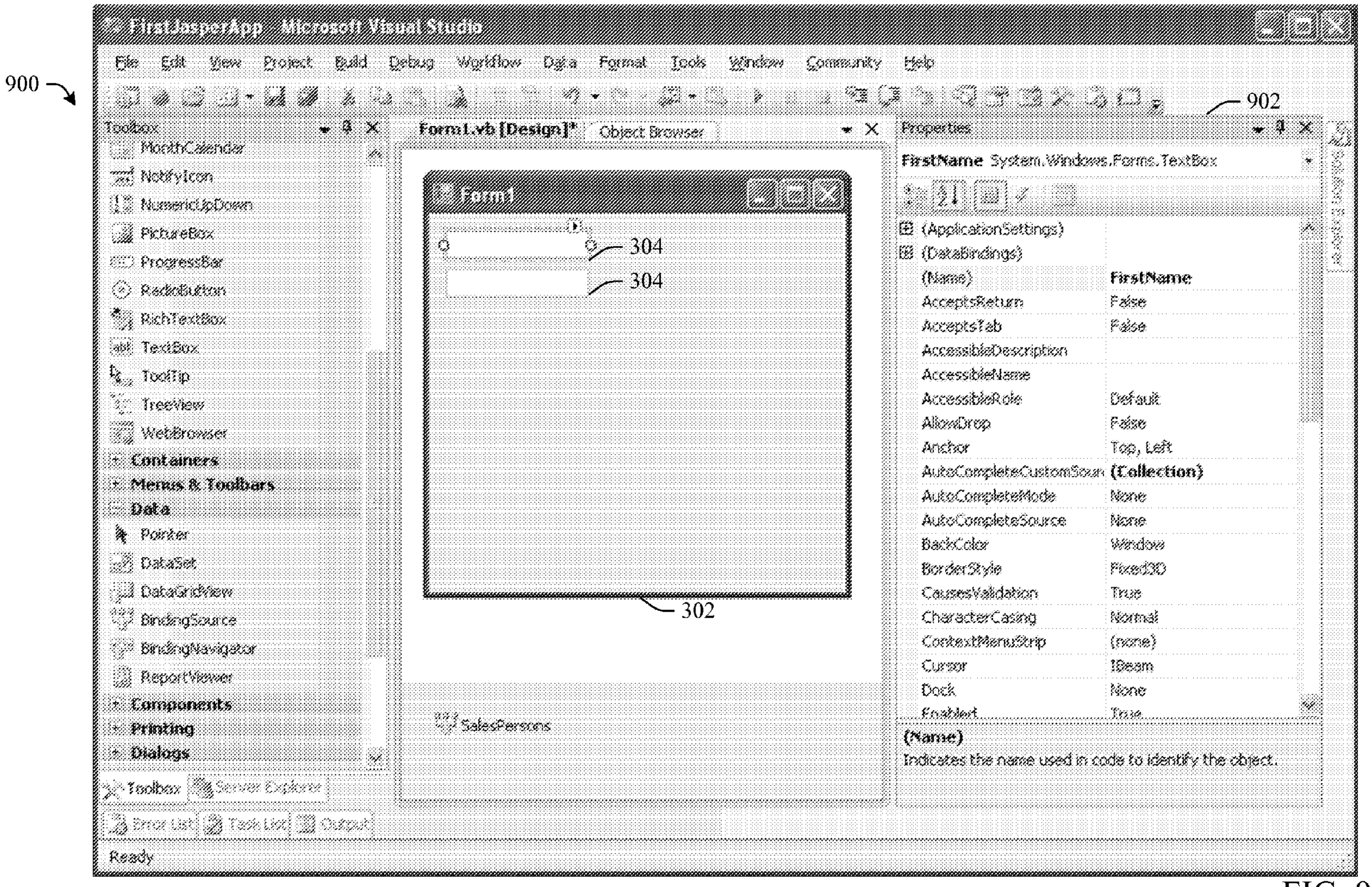

900
FirstJasperApp - Microsoft Visual Studio
File Edit View Project Build Debug Workflow Data Format Tools Window Community Help
902
Toolbox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
WebBrowser
Containers
Menus & Toolbars
Data
Pointer
DataSet
DataGridView
BindingSource
BindingNavigator
ReportViewer
Components
Printing
Dialogs
Form1.vb [Design]*
Object Browser
Form1
304
304
302
SalesPersons
Properties
FirstName System.Windows.Forms.TextBox
(ApplicationSettings)
(DataBindings)
(Name) FirstName
AcceptsReturn False
AcceptsTab False
AccessibleDescription
AccessibleName
AccessibleRole Default
AllowDrop False
Anchor Top, Left
AutoCompleteCustomSour (Collection)
AutoCompleteMode None
AutoCompleteSource None
BackColor Window
BorderStyle Fixed3D
CausesValidation True
CharacterCasing Normal
ContextMenuStrip (none)
Cursor IBeam
Dock None
Enabled True
(Name)
Indicates the name used in code to identify the object.
Toolbox
Server Explorer
Error List
Task List
Output
Ready

FIG. 9

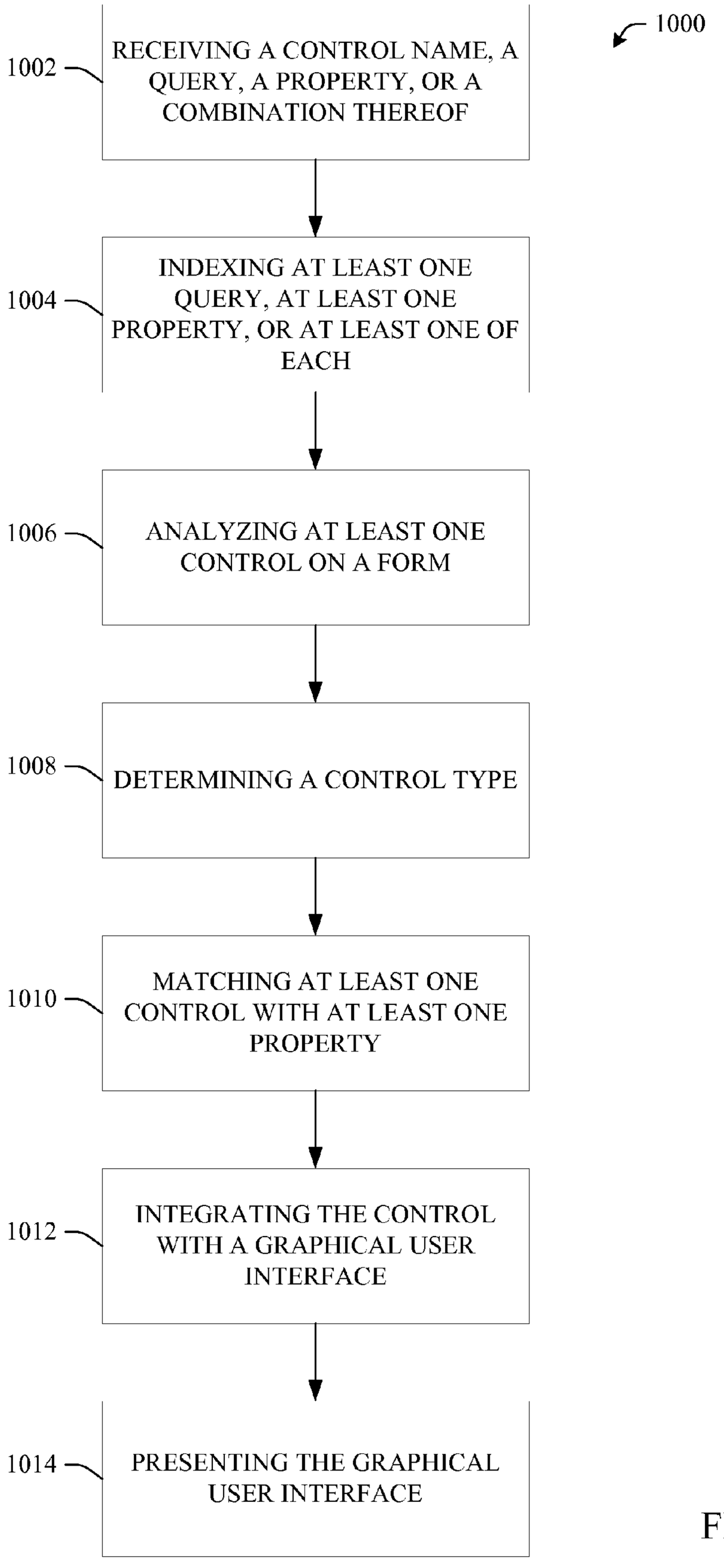

1000
1002
RECEIVING A CONTROL NAME, A QUERY, A PROPERTY, OR A COMBINATION THEREOF
1004
INDEXING AT LEAST ONE QUERY, AT LEAST ONE PROPERTY, OR AT LEAST ONE OF EACH
1006
ANALYZING AT LEAST ONE CONTROL ON A FORM
1008
DETERMINING A CONTROL TYPE
1010
MATCHING AT LEAST ONE CONTROL WITH AT LEAST ONE PROPERTY
1012
INTEGRATING THE CONTROL WITH A GRAPHICAL USER INTERFACE
1014
PRESENTING THE GRAPHICAL USER INTERFACE

FIG. 10

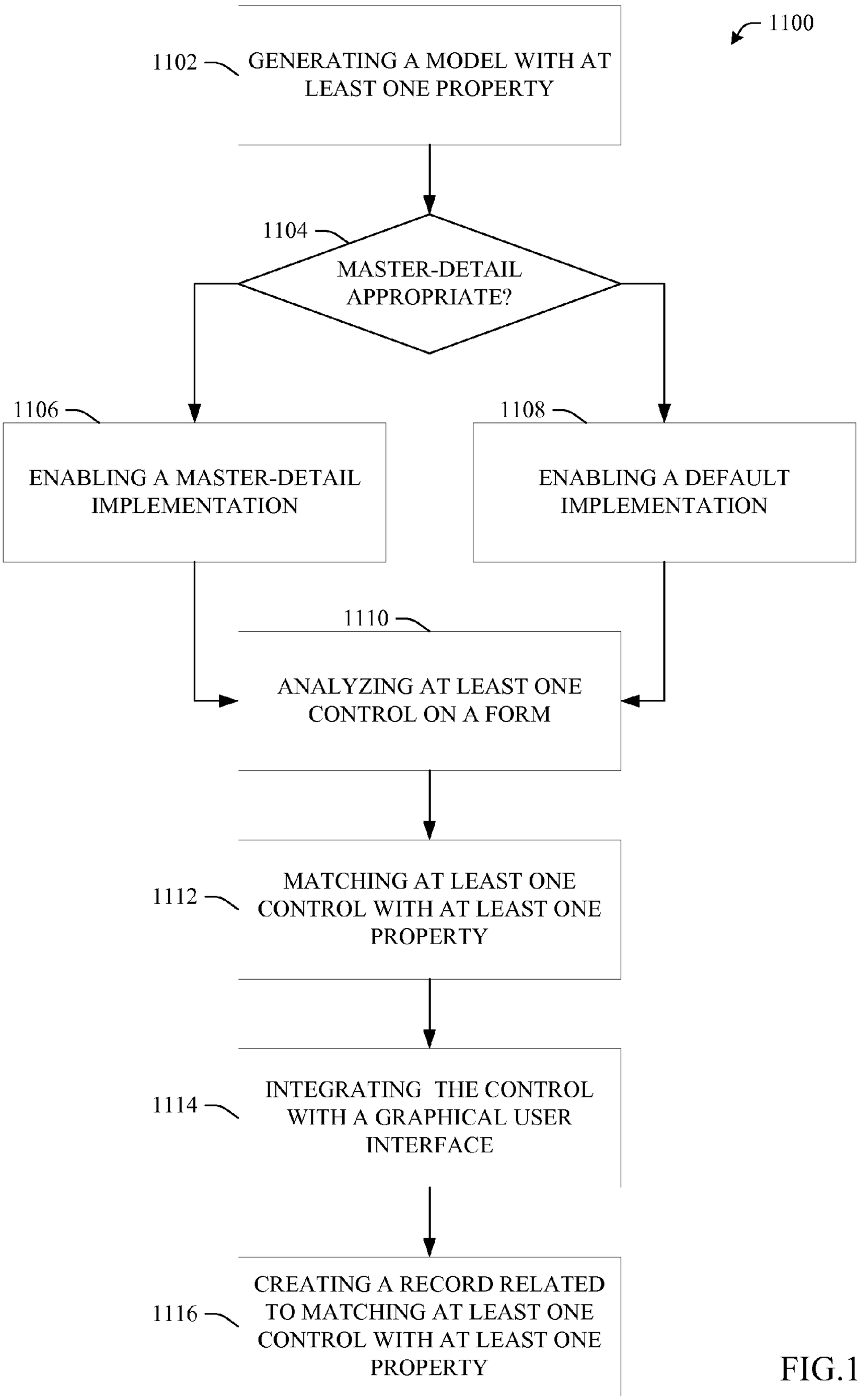

1100
1102
GENERATING A MODEL WITH AT LEAST ONE PROPERTY
1104
MASTER-DETAIL APPROPRIATE?
1106
ENABLING A MASTER-DETAIL IMPLEMENTATION
1108
ENABLING A DEFAULT IMPLEMENTATION
1110
ANALYZING AT LEAST ONE CONTROL ON A FORM
1112
MATCHING AT LEAST ONE CONTROL WITH AT LEAST ONE PROPERTY
1114
INTEGRATING THE CONTROL WITH A GRAPHICAL USER INTERFACE
1116
CREATING A RECORD RELATED TO MATCHING AT LEAST ONE CONTROL WITH AT LEAST ONE PROPERTY

FIG.11

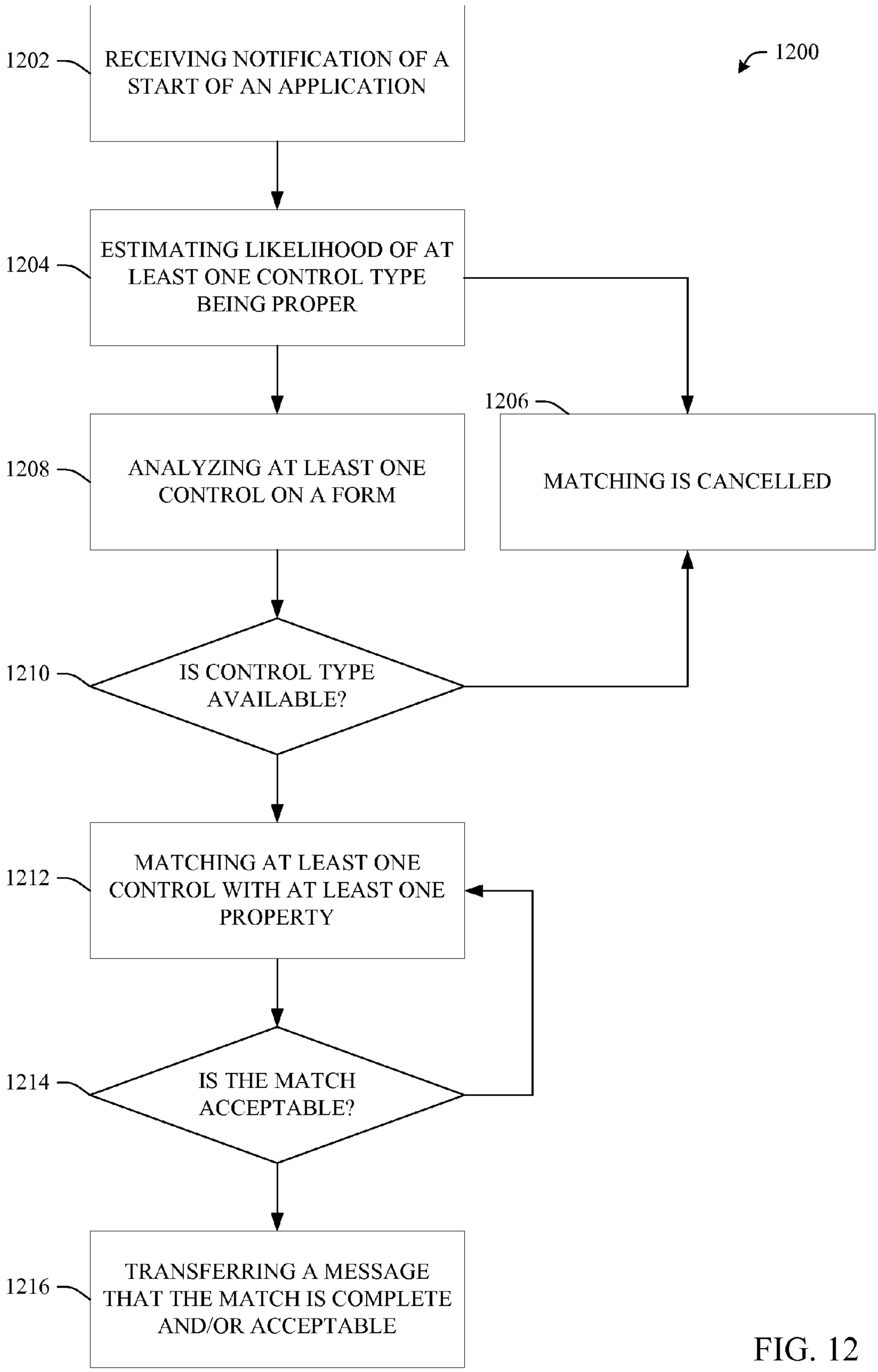
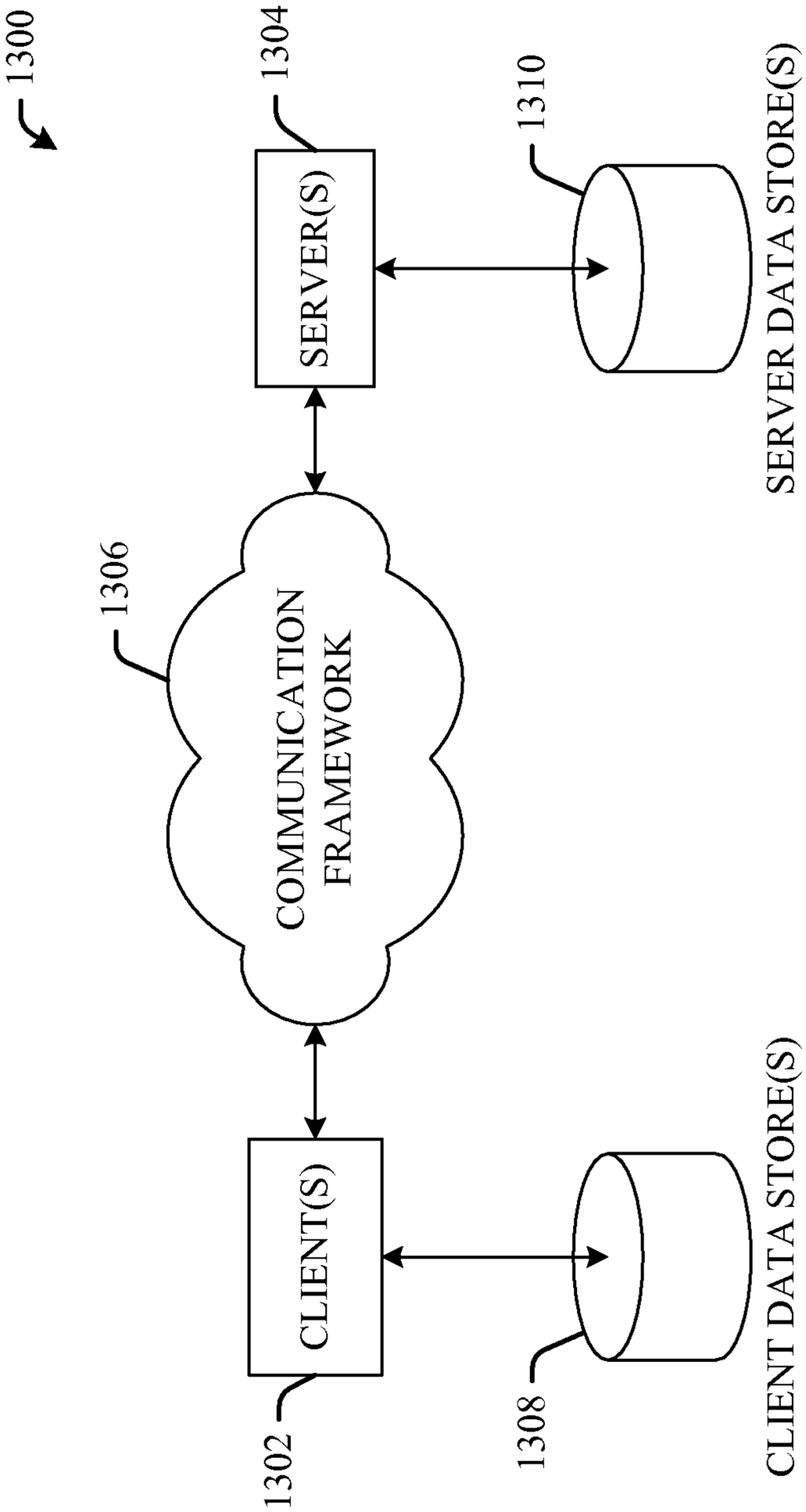

1200
1202
RECEIVING NOTIFICATION OF A START OF AN APPLICATION
1204
ESTIMATING LIKELIHOOD OF AT LEAST ONE CONTROL TYPE BEING PROPER
1206
MATCHING IS CANCELLED
1208
ANALYZING AT LEAST ONE CONTROL ON A FORM
1210
IS CONTROL TYPE AVAILABLE?
1212
MATCHING AT LEAST ONE CONTROL WITH AT LEAST ONE PROPERTY
1214
IS THE MATCH ACCEPTABLE?
1216
TRANSFERRING A MESSAGE THAT THE MATCH IS COMPLETE AND/OR ACCEPTABLE

FIG. 12

FIG. 13


1300
1304
SERVER(S)
1310
SERVER DATA STORE(S)
1306
COMMUNICATION FRAMEWORK
1302
CLIENT(S)
1308
CLIENT DATA STORE(S)

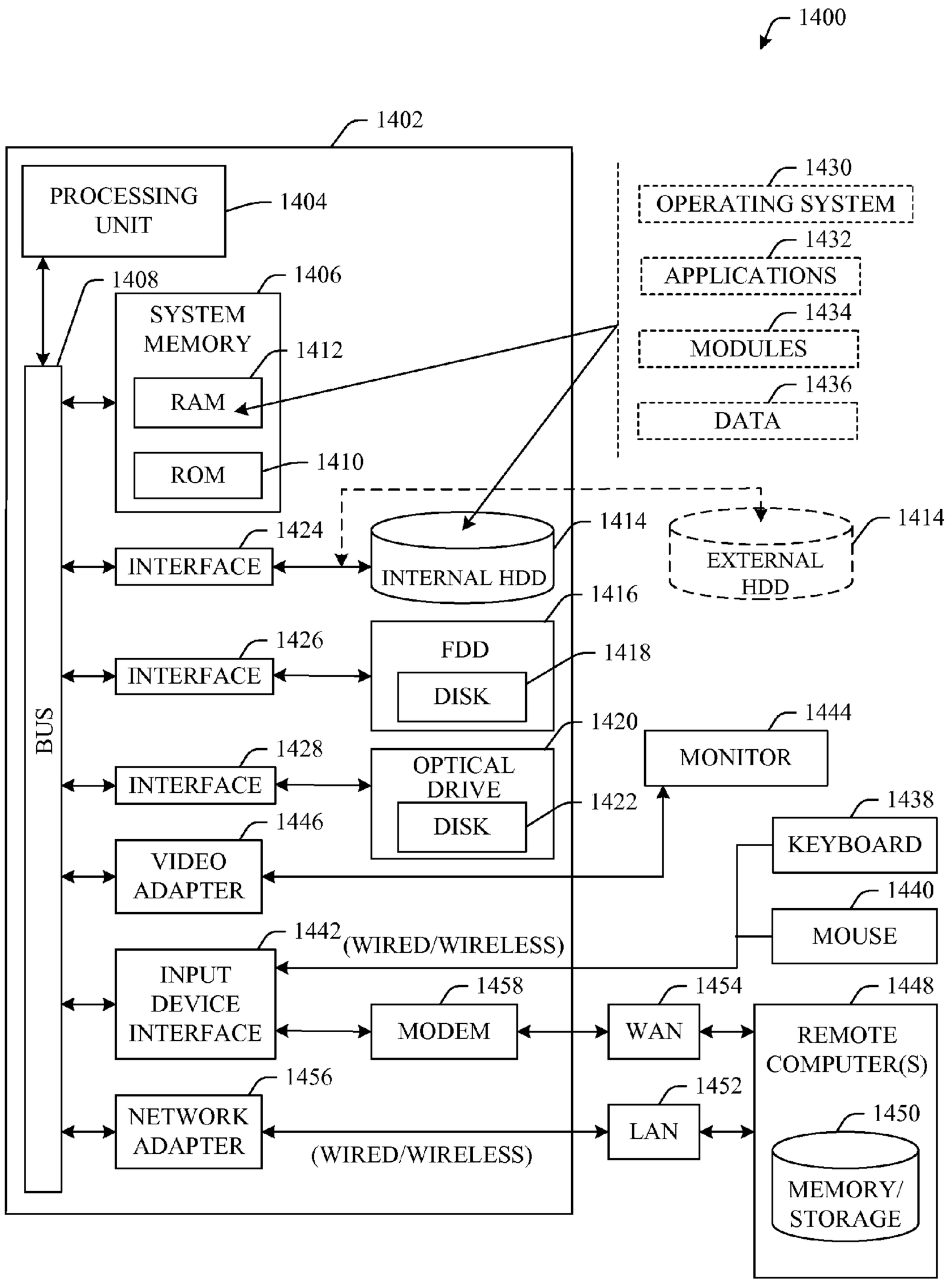

1400
1402
PROCESSING UNIT
1404
1408
1406
SYSTEM MEMORY
1412
RAM
ROM
1410
BUS
1424
INTERFACE
1414
INTERNAL HDD
1414
EXTERNAL HDD
1426
INTERFACE
1416
FDD
DISK
1418
1428
INTERFACE
1420
OPTICAL DRIVE
DISK
1422
1446
VIDEO ADAPTER
1442
(WIRED/WIRELESS)
INPUT DEVICE INTERFACE
1458
MODEM
1456
NETWORK ADAPTER
(WIRED/WIRELESS)
1430
OPERATING SYSTEM
1432
APPLICATIONS
1434
MODULES
1436
DATA
1444
MONITOR
1438
KEYBOARD
1440
MOUSE
1454
WAN
1452
LAN
1448
REMOTE COMPUTER(S)
1450
MEMORY/ STORAGE

FIG. 14

AUTOMATIC RUNTIME CONTROL BINDING

CROSS-REFERENCE

[0001] This application claims priority to U.S. Provisional Application Ser. No. 60/913,183 entitled "AUTOMATIC DATABINDING BINDING OF DYNAMICALLY/RUNTIME GENERATED TYPES TO FORMS" filed on Apr. 20, 2007. The entirety of which is incorporated by reference herein.

[0002] This application also claims priority to U.S. Provisional Application Ser. No. 60/913,180 entitled "GENERATION OF RUNTIME TYPES AND EXTENSIBILITY OF NAME GENERATION" filed on Apr. 20, 2007. The entirety of which is incorporated by reference herein.

[0003] This application additionally claims priority to U.S. Provisional Application Ser. No. 60/913,186 entitled "TRANSLATING LATE BOUND LINQ EXPRESSIONS INTO DATABASE QUERIES" filed on Apr. 20, 2007. The entirety of which is incorporated by reference herein.

TECHNICAL FIELD

[0004] The subject specification relates generally to binding controllers to data and in particular to automatic binding of controllers to data at runtime.

BACKGROUND

[0005] Computers have become an important part of daily life for many individuals. Computers influence a wide range of activities from business transactions (e.g., wire-to-wire bank transactions) to interpersonal communications (e.g., communication through text messaging.) As computers become more popular, different mechanisms become available to allow users with minimal background in computers to obtain a relatively high level of functionality.

[0006] One area of development that has enabled computers to become more prevalent in society a graphical user interface (GUI.) GUIs can be utilized in an array of personal electronic devices, such as personal computers, cellular telephones, personal digital assistants, etc. A conventional computer can employ a keyboard, monitor, and mouse that a user can engage. A user is presented a GUI through the monitor by performing an operating on a keyboard (e.g., typing characters that start an application that uses a GUI.) The user can interact with the GUI through contacting the GUI with a mouse pointer that displays on the screen that is controlled by the mouse.

[0007] Different GUI capabilities allow different users to have a customizable experience. For example, if a child is operating an electronic device, then the GUI can disclose recognizable cartoon characters that the child will identify. Moreover, a user can modify a GUI to match their personal style. An individual can have a favorite color that permeates a GUI and/or have a theme that is used throughout the GUI (e.g., puppies.) In addition, a GUI can disclose animations that assist a user in navigating a program (e.g., a flashing red light indicates to the user that they should stop an activity.)

SUMMARY

[0008] The following discloses a simplified summary of the specification in order to provide a basic understanding of some aspects of the specification. This summary is not an extensive overview of the specification. It is intended to neither identify key or critical elements of the specification nor delineate the scope of the specification. Its sole purpose is to disclose some concepts of the specification in a simplified form as a prelude to the more detailed description that is disclosed later.

[0009] Conventionally, two main methods are employed in binding a control to source information. In a first method, a developer writes numerous lines of code to create individual bindings; however, this code can be highly error prone and become tedious to the developer. In a second method, a tool (e.g., wizard) is presented at design time to a user to lead the user through binding generation. Nonetheless, the wizard can create a number of difficulties. If a programmer wants to have controls that display properties of classes that are not generated until runtime, then a typical wizard will not achieve desired functionality because properties are not known since a class does not yet exist. Additionally, a typical wizard looks at a snapshot of a data source based on when the wizard is run. This can cause difficulties when developers are building an application over time and wizard generated information becomes out of date with schemas of a data store.

[0010] The subject specification discloses information that relates to automatic binding generation at runtime. At design time, computer code is received that adds controls, a binding source, and/or properties relating to what is to be reflected in the control. At runtime, a system automatically creates at least one binding based on received computer code. Therefore, bindings can be created by a developer though writing a significantly smaller amount of code; thus the developer can operate more efficiently (e.g., write less) and write more successfully (e.g., writing less code can create a lower likelihood of error.)

[0011] Various features can integrate with automatic binding creation at runtime. A binding can integrate with a graphical user interface that allows a user to achieve a relatively high level of functionality though the binding. Moreover, a binding can be constructed from a model is generated from a database; bindings are created via the model and generated types. To assist in creating a fluent binding structure, a user can employ a similar naming convention as is used by a database. Different configurations are available to a developer, such as integrating a control with a Boolean operator to disclose if something is true/false, aggregating information from different source, etc.

[0012] The following description and the annexed drawings set forth certain illustrative aspects of the specification. These aspects are indicative, however, of but a few of the various ways in which the principles of the specification may be employed. Other advantages and novel features of the specification will become apparent from the following detailed description of the specification when considered in conjunction with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIG. **1** illustrates a representative autobinding system in accordance with an aspect of the subject specification.

[0014] FIG. **2** illustrates a representative autobinding system with a display component in accordance with an aspect of the subject specification.

[0015] FIG. **3** illustrates a representative binding configuration with auxiliary units in accordance with an aspect of the subject specification.

[0016] FIG. **4** illustrates a representative autobinding system with auxiliary components in accordance with an aspect of the subject specification.

[0017] FIG. **5** illustrates a of a representative search component in accordance with an aspect of the subject specification.

[0018] FIG. **6** illustrates a of a representative autobinding system with two applications in accordance with an aspect of the subject specification.

[0019] FIG. **7** illustrates a of a representative autobinding system with an aggregation capability in accordance with an aspect of the subject specification.

[0020] FIG. **8** illustrates a of a representative context class construction methodology in accordance with an aspect of the subject specification.

[0021] FIG. **9** illustrates a representative autobinding screenshot in accordance with an aspect of the subject specification.

[0022] FIG. **10** illustrates a representative rule autobinding creation methodology in accordance with an aspect of the subject specification.

[0023] FIG. **11** illustrates a representative autobinding building methodology with a master-detail configuration in accordance with an aspect of the subject specification.

[0024] FIG. **12** illustrates a representative autobinding generation methodology with verification actions in accordance with an aspect of the subject specification.

[0025] FIG. **13** illustrates an example of a schematic block diagram of a computing environment in accordance with the subject specification.

[0026] FIG. **14** illustrates an example of a block diagram of a computer operable to execute the disclosed architecture.

DETAILED DESCRIPTION

[0027] The claimed subject matter is now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the claimed subject matter. It may be evident, however, that the claimed subject matter may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to facilitate describing the claimed subject matter.

[0028] As used in this application, the terms "component," "module," "system", "interface", or the like are generally intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. For example, a component may be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a controller and the controller can be a component. One or more components may reside within a process and/or thread of execution and a component may be localized on one computer and/or distributed between two or more computers. As another example, an interface can include I/O components as well as associated processor, application, and/or API components.

[0029] Furthermore, the claimed subject matter may be implemented as a method, apparatus, or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof to control a computer to implement the disclosed subject matter. The term "article of manufacture" as used herein is intended to encompass a computer program accessible from any computer-readable device, carrier, or media. For example, computer readable media can include but are not limited to magnetic storage devices (e.g., hard disk, floppy disk, magnetic strips . . .), optical disks (e.g., compact disk (CD), digital versatile disk (DVD) . . .), smart cards, and flash memory devices (e.g., card, stick, key drive . . .). Additionally it should be appreciated that a carrier wave can be employed to carry computer-readable electronic data such as those used in transmitting and receiving electronic mail or in accessing a network such as the Internet or a local area network (LAN). Of course, those skilled in the art will recognize many modifications may be made to this configuration without departing from the scope or spirit of the claimed subject matter.

[0030] Moreover, the word "exemplary" is used herein to mean serving as an example, instance, or illustration. Any aspect or design described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other aspects or designs. Rather, use of the word exemplary is intended to disclose concepts in a concrete fashion. As used in this application, the term "or" is intended to mean an inclusive "or" rather than an exclusive "or". That is, unless specified otherwise, or clear from context, "X employs A or B" is intended to mean any of the natural inclusive permutations. That is, if X employs A; X employs B; or X employs both A and B, then "X employs A or B" is satisfied under any of the foregoing instances. In addition, the articles "a" and "an" as used in this application and the appended claims should generally be construed to mean "one or more" unless specified otherwise or clear from context to be directed to a singular form.

[0031] FIG. **1** discloses an example system **100** for creation of an automatic bind at runtime. Since classes can be generated at runtime (e.g., during operation of an application after compile time), it can be difficult to manually create bindings prior to the existence of a class (e.g., at design time.) The system **100** allows bindings to be created automatically through addition of relatively minimal code by a developer (e.g., commonly one to two lines of code.)

[0032] A communication component **102** engages other units and transfers data between the system **100** and at least one other unit. For instance, the communication component **102** can receive signals that runtime has commenced, that a control that should be bound, as to what type of control should be bound, as to what information should bind with a control, etc. Some data can be received by reading storage that holds computer code written by a developer.

[0033] Moreover, the communication component **102** obtains data that relates to runtime operation of an application. Example data that is obtained is that runtime has commenced and that other components should start operation, what control is to be bound, what control type is to be bound (e.g., ListBox, Textbox, etc.), what source of information is to be bound, etc. In addition, information can be obtained and transferred through a single transaction.

[0034] The communication component **102** can operate according to a number of different embodiments. According to one configuration, the communication component **102** operates wirelessly with other units through encrypted and/or non-encrypted manners. Hard wires transmissions and receptions can be made by the communication component **102** also using encrypted/non-encrypted manners. The communication component **102** can employ different contact techniques with different units. For instance, a first unit can be engaged

through wireless encryption while a second unit is appointed through wired, non-encrypted interaction.

[0035] A bind component **104** collects information from the communication component **102**. The bind component creates a link (e.g., binding **106**) between a controller **108** and a source of information **110** based on obtained data from the communication component **102**. Example information from the communication component **102** that can be employed by the bind component **104** is a signal that runtime has begun and thus, the bind component **104** should commence operation, what control and/or source information should be bound, if a request is authorized, etc. To create a link, the bind component **104** can use a name convention (e.g., names entered through a programming language match names in an entity.)

[0036] The bind component can embrace the control **108** and the source of information **110**. Through the embrace, a binding can be set up between the two of them, such that the control **108** discloses information that relates to the source **110**. While the control **108** discloses a portion of source information, it can actually link to the entire source (e.g., object, class, table, etc.) A control **108** can be a visual element capable of displaying information. The bind component **104** can include various check capabilities, such as determining if the binding **106** between the control **108** and the source of in formation **110** is successful.

[0037] In addition, the bind component **104** can engage the communication component **102** to disclose information that relates to the binding **106**. For instance, once a proper binding **106** is established, the bind component **104** utilizes the communication component **102** to notify an auxiliary component that the binding was successfully completed. Moreover, the bind component **104** can use the communication component **104** to discover further actions. For instance, a bind component **104** can receive an instruction that a binding **106** is to be created. Once the binding is complete, the bind component **104** sends a message to the communication component **102** signifying completion and the communication component **102** requests a second binding to be made. It is to be appreciated that bind component **104** can operate concurrent bindings at one time (e.g., create two bindings at one time.)

[0038] FIG. **2** discloses an example system **200** for creating a binding automatically with utilization of a display component **202**. A communication component **102** obtains data that relates to runtime operation of an application. A bind component **104** creates a link (e.g., binding **106**) between a control **108** and information based at least in part on obtained data. The bind component **104** can use an analysis component **204** to examine a model **206** and bind the model **108** (e.g., a portion of the model) with at least one control **108**. A model **206** can be a representation of stored information, such as information saved in a database. The bind component **104** can function as a means for binding the model **206** to at least one control.

[0039] The system **200** can engage a display component **202** to disclose information related to the binding **106**. The display component **202** presents at least one control linked with information through the bind component **104** and can manifest as a monitor, projector, hologram presentment component, etc. A form can be presented to a user that includes a control that links to an information source (e.g., the model **206**.) Moreover, the display component **202** can facilitate integration of the control **108** with a graphical user interface (GUI) that allows a user to interact with the control **108**. For instance, the control **108** can be a combobox (e.g., a control that has multiple entries that discloses one entry as a time and allows a user to scroll to other entities). A user can modify the combobox through engagement of the GUI (e.g., clicking on an arrow.) In addition, the display component **202** can present to a developer what was matched as well as what was not matched. The display component **202** can show to a developer at least two details are disclosed, where a first detail discloses at least one link that was created and a second detail discloses at least one link that was intended to be created, but was not created (e.g., code that was written to provide a binding, but because of an error the binding did not occur.)

[0040] The model **206** can be mapped to a graphical user interface through the display component **202**. According to one embodiment, there is generation of a model **206** to represent a configuration (e.g., different tables of a database as well as relationships between the different tables.) The model **206** is generated from a backend and a binding is performed upon the model and types. The analysis component **204** performs various types of investigation upon the model **206** to determine at least one appropriate binding. For instance, the analysis component **204** can identify relationships within the model **206**, search appropriate information for binding, etc. Moreover, the analysis component **204** can operate upon units other then the model **206**. For instance, an examination can take place of the communication component **102**, control **108**, display component **202**, etc.

[0041] The analysis component **204** can determine at least one asset related to a control **108**. Example assets are a control type (e.g., listbox, combobox, checkbox, textbox, etc.), if a control is capable of supporting a link, security clearance of a control, etc. The analysis component **204** can function as a means for analyzing a model **206**. While the subject specification discloses the analysis component **204** as integrating with the bind component **104**, the analysis component **204** can operate independent of the bind component **104** and engage in communication with devices in addition to integrating with other devices (e.g., a processor.).

[0042] The system **200** can also employ a master-detail implementation, commonly through the bind component **104**. A master-detail implementation allows for further binding to show relationships between different entities in a model **206**. In an example master-detail implementation, a form holding a control **108** that displays customers exists, but there is also a desire to bind orders automatically that relate to the customer. The customer can be the master record while the orders are considered detail.

[0043] FIG. **3** discloses an example multiple binding configuration **300** with a display component **202**. A display component **202** can disclose a form **302** with two or more different controls **304***a* and **304***b*. The controls **304***a* and **304***b* can bind with information from an entity **306**. Commonly, an entity is a portion of a model **206** of FIG. **2**; however, it is possible that an entity be a stand-alone unit.

[0044] An entity can have a number of pieces of information **308***a* to **308***d* that can be displayed through a control. The pieces of information **308** (e.g., properties) can have a corresponding identifier **310***a* to **310***d* that enables communication of what is contained in a piece of information. An identifier **310** can be used by a bind component **104** of FIG. **1** to ascertain quickly a piece of information **308**. Different pieces of information **308** can have different relationships with one another. For instance, if information **306***c* is blank, then automatically information **306***d* can convert to a blank since no quantity can indicate that there is no product for purchase.

[0045] A bind component **104** of FIG. **1** can receive an instruction that a form **302** on a display component **202** should disclose two textboxes as controls **304***a* and **304***b*. The bind component creates bindings **312** and **314** between the entity **306** and the form **302**. Specifically, bindings **312** and **314** are between requested controls **304***a* and **304***b* and pieces of information **308***a* and **308***d* respectively. If information in entity **306** changes (e.g., a customer adds a middle name), then the change can be reflected in a corresponding controls (e.g., control **304***a*.)

[0046] In a specific example, the display component **202** can be a monitor for a shipping company. A form **302** is displayed on the monitor to disclose what a customer purchases and the name of the customer. Two controls **304** are utilized, a first control **304***a* shows a customer name and a second control **304***b* discloses a purchased product. The bind component **104** of FIG. **1** identifies from written code that at runtime a binding **312** should be created between control **304***a* and information **306***a* as well as a binding **314** between control **304***b* and information **306***a*. To help guide the bind component **104** of FIG. **1**, identifiers **310** can be used that disclose where information is located. For instance, programming language uses a term 'Product' and the bind component **104** of FIG. **1** can look for a term 'Product' in the entity.

[0047] FIG. **4** discloses an example system **400** for creating a binding through use of various components. A bind component **104** can perform an analysis of an available control located on the form **302** (e.g., through use of an analysis component **204** of FIG. **2**.) This can be done through an analysis component **204** of FIG. **2**. As part of the analysis, the bind component **104** can determine how many controls are to be bound, capabilities and limitations of a control, etc.

[0048] A search component **402** performs an examination upon at least one query **404**. The examination can include possible returns of a query when it is run upon a database. While a general query is disclosed, a query can actually contain a number of sub-queries (e.g., customers query, orders query, product query, etc.) Moreover, the search component **402** can analyze properties **406** that relate to the query **404**. For example, a query can disclose a table in a database named 'shipping' where the table holds properties 'product', quantity', 'address', etc.

[0049] An add component **408** can place results of examination of at least one query into a bind source index **410**. The bind component **104** can access the bind source index **410** to determine what is available for linkage. For instance, an examination can state that a table 'customer' is a result of the query **404**. The add component **408** places a record in the bind source index **410** that table 'customer' is available for binding. When the bind component **104** operates and makes a request to the bind source index **410** through use of the communication component **102**, the bind component **104** learns that the table 'customer' is available (e.g., the communication component **102** transfers a result to the bind component **104**.) The add component **408** populates an index with at least one piece of data during runtime, where the communication component **102** obtains at least one piece of index data that transfers to a bind component.

[0050] The bind source index **410** can include both a source index **412** and an item index **414**. There can be multiple instances where different information is located on different sources. The source index **412** allows information to be organized based on where information is located. As an illustrative example, a form **302** can hold two controls, one is to disclose an individual's security clearance and another is to present a person's home telephone number. It is likely security information is held in one location (e.g., an encrypted database) while telephone information is detained in a different location (e.g., digital directory.) The source index **412** allows the bind component **104** to ascertain quickly where information is located.

[0051] The source index **412** can work in conjunction with an item index **414**. While the source index **412** holds information that relates to location, the item index **414** can hold what information pieces are in different locations. This can provide a bind component **104** with a clear picture as what is available when accessing a bind source index **410**. Based on information found in the bind source index **410**, the bind component **104** can make proper bindings.

[0052] FIG. **5** discloses an example search component **402** that can operate with other component disclosed in the subject specification. The search component **402** performs an investigation to locate data for obtainment by the communication component **102** of FIG. **1**. A transmission component **502** allows the search component **402** to interact with other units. The transmission component **502** can engage in both sending out transmissions as well as receiving transmissions. Various configurations of the transmission component **502** can be used, including wireless communication, wired engagement, encrypted and/or non-encrypted announcements, etc.

[0053] An authentication component **504** can perform different types of verification concerning the search component **402**. According to one embodiment, the authentication component **504** confirms that a control can have access to information. For instance, a user can write code for a control to display a password for 'John_Q'. However, the user does not have authorization to view that information. Therefore, the authentication component **504** can block the bind component **104** of FIG. **1** from creating a link and thus prevent the user from obtaining the password.

[0054] The authentication component **504** can configure to determine if at least one error occurs in relation to the search component **502**. For instance, signal can transfer to the search component **402** that a search component is to examine a query. However, runtime has not commenced and a class the query is to run on has not yet been constructed. Therefore, the authentication component **502** can send notification through the transmission component **502** that an error has taken place (e.g., an improper operation.) Moreover, the authentication component **504** can include capabilities that allows for error identification and/or correction (e.g., finding a source of an error, repairing a module in error, etc.) The authentication component **504** determines if operation of the bind component **104** of FIG. **1** is proper (e.g., the bind component **104** of FIG. **1** can function without an error, the bind component **104** of FIG. **1** is authorized to create a requested binding, a diagnostic standard is met, etc.)

[0055] Storage **506** can be used to assist modules in a system with the search component **402** as well as the search component **402**. Various components can create and access records of the storage **506**. For instance, logic used by the authentication component **504** to determine if a user has authorization to write code to gain password information can be held in storage **506**. Furthermore, storage **506** can have various types of security protection (e.g., a list of authorized components.) Storage **506** retains at least one record that relates to operation of the bind component **104** of FIG. **1** (e.g.,

holds the binding source index **410**, retains security information, includes logic used by the authentication component, provides data for obtainment to the communication component **102** of FIG. **1**, etc.)

[0056] An artificial intelligence component **508** makes at least one inference or at least one determination or at least one of each in relation to creation of a link between a control and information. Various scenarios can occur that are processed by the artificial intelligence component **508**. For example, there can be multiple indexes accessible to a bind component **104** of FIG. **1**. Artificial intelligence component **508** can make a determination as to what index likely contains a path to an appropriate data source. A determination result can transfer to a bind component **104** through utilization of a transmission component **502**.

[0057] The artificial intelligence component **508** can employ one of numerous methodologies for learning from data and then drawing inferences and/or making determinations related to automatic bind creation at runtime (e.g., Hidden Markov Models (HMMs) and related prototypical dependency models, more general probabilistic graphical models, such as Bayesian networks, e.g., created by structure search using a Bayesian model score or approximation, linear classifiers, such as support vector machines (SVMs), non-linear classifiers, such as methods referred to as "neural network" methodologies, fuzzy logic methodologies, and other approaches that perform data fusion, etc.) in accordance with implementing various automated aspects described herein. Methods also include methods for the capture of logical relationships such as theorem provers or more heuristic rule-based expert systems.

[0058] A query component **510** determines available queries in a context and performs a requested query. For instance, computer code generated by a user at design time can provide a particular context where information is located. An example context runs a customer query, an order query, and a product query. A property component **512** locates properties that result form a query. A property can be an individual piece of information (e.g., in a table 'customer' returned by a 'customer' query, a property is an entry 'John_Q'.) An index component **514** processes results from the query and selects what data should be populated into the binding source index **410** by the add component **408**. A form component **516** locates controls that are to be expected to integrate with an autobinding; commonly, the controls are on a form. Information obtained by the query component **510**, property component **512**, index component **514**, and/or form component **516** can transfer to the communication component **102** of FIG. **1** for operation by the bind component **104** of FIG. **1** though the transmission component **502**.

[0059] FIG. **6** discloses an example multiple application configuration **600** with autobinding at runtime. A single control can include binds to information from different applications. A communication component **102** obtains data that relates to runtime operation of at least one application. A bind component **104** creates a link between a control and information based at least in part on obtained data. A display component **202** can hold two controls **602** and **604** on a form or independent of a form.

[0060] Control **602** can have a binding **606** with an online application **608**. Information from the online application **608** can change and the control **602** can reflect an appropriate change. Control **604** links through a binding **610** to an internal application **612**. At runtime of application **608** and/or **612**, bindings can be created; thus, bindings do not have to occur in the same display component **202**/form at one time (e.g., a first binding can be created prior to creation of a second binding.) In the configuration **600**, the bind component **104** creates at least two links, wherein at least one link integrates with a first application (e.g., online application **608**) and at least one link integrates with a second application (e.g., internal application **612**.)

[0061] In an example implementation, a company can allow orders for products to be placed according to two different manners. An online webpage runs an application that allows users to place orders. In addition, purchasers can call into a telephone center and a receptionist at a computer terminal enters orders into a database. A single display can present linked information through binding data from different sources (e.g., an online source and internal source.)

[0062] FIG. **7** discloses an example autobinding configuration **700** with utilization of an aggregation component **702**. A single control **108** can include binding **106** to information **704** from different applications. A communication component **102** obtains data that relates to runtime operation of at least one application. A bind component **104** creates a link between the control **108** and information **704** based at least in part on obtained data.

[0063] The aggregation component **702** can combine information from at least two different information sources **706** and **708**. The sources **706** and **708** can be different applications or different portions of a single application. The aggregation component **702** combines information from the two sources and creates a common piece of information **704** that can obtain a binding **106** through the bind component **104**. The aggregation component **702** can operate according to a number of different configurations. For instance, the aggregation component **702** can add data together from different sources, streamline data from different sources (e.g., remove redundant entries.), etc. The aggregation component **702** builds information **704** based on material from at least two sources **706** and **708**, where the information **704** is provided to the bind component **104** for linkage (e.g., performed through the communication component **102**.)

[0064] For example, a company can run an online store and a mail order store that sell the same products. A user 'John_Q' can purchase two items online and three items through mail order. A developer can have a desire to have a control reflect the total amount of items purchased by 'John_Q'; however, there is not one source that provides that information. The aggregation component **702** can receive a notice from the communication component **102** that this is what is desired from the developer. The aggregation component **702** reads the item amounts from the sources **706** and **708** and combines them into one piece of information **704**. Therefore, an automatic binding can be created at runtime according to desires of a developer.

[0065] FIG. **8** discloses an example autobinding configuration **800**. A form **302** can include two different fields **802** that represent controls. The form **302** can integrate with at least one binding source **804** that enables interaction between a data source (e.g., dynamic context) and a control that binds with data. The fields can integrate with different rules **806** that allows for different field implementations. These rules can include a listbox (e.g., single entity), combobox (e.g., multiple entries), checkbox (e.g., flag that something is true with Boolean operator), textbox (e.g., display text that something

is false), etc. Two indexes **808** can be used to provide information to the form **302** for binding with a field **802**.

[0066] FIG. **9** discloses an example screenshot **900** of a presentation for creating autobinding at runtime. A form **302** presents to user that has an ability to include at least one control **304**. A user can select from a list of possible entries **902** that allow the user to cerate an appropriate form **302**. The user can employ a 'drag-and-drop' function to cerate the desired form **302** by engaging an entry and placing its graphical representation upon the form **302**. After a user completes operating a program disclosed from the screenshot, then systems, methodologies, configurations, etc. of the subject specification automatically create at least one binding between desired elements.

[0067] FIG. **10** discloses a methodology **1000** for creating binding automatically at runtime. Action **1002** is receiving a control name, a query, a property, or a combination thereof. Various amounts of received information can assist in creating a binding automatically at runtime. A control name allows for identification of what will bind with information. Reception of a query allows for knowledge of what was searched for at a source (e.g., database) and enables an improved tracking of overall events. A property is commonly a piece of information that will bind with a control. Reception of these items can (e.g., control name, query, and/or property) can include storage of the items.

[0068] There is indexing at least one query, at least one property, or at least one of each **1004**. Indexing allows data to be organized and to be gathered efficiently (e.g., quickly and with a relatively small chance for error.) Indexing commonly includes placing the information in a set storage location that can be accessed by various units operating at least portions of the methodology **1000**.

[0069] Analyzing at least one control on a form **1006** occurs. Analysis can include a number of different examinations. One common examination is determining if a control is capable of receiving a proper bind. For example, a control can have a relatively small display area while information that is intended to be bound is relatively large. Since this control could a poor choice for binding with intended information, this can be beneficial information. According to one embodiment, analyzing a control on a form is locating the control. Other actions that are non-exhaustive examples of analysis are determining if a control is already bound, investigating a form to determine what controls are on the form, performing a virus check on a control, etc.

[0070] Event **1008** is determining a control type. A control can take different configurations and engage in different types. The control can be hard coded (e.g., the control is not able to change) or loose (e.g., the methodology **1000** can select a rule for a control and provide a control with a type for a binding.) Example control types are listbox (e.g., single entity), combobox (e.g., multiple entries), checkbox (e.g., flag that something is true with Boolean operator), textbox (e.g., display text that something is false), etc.

[0071] There is matching a least one control with at least one property **1010**. Matching a control with a property creates a binding between them and this can take place automatically at runtime. Matching can include various verification actions. For instance, a control can already have a binding (e.g., be configured to display another property of another application.) Event **1010** can make this determination and match the property with another control. Moreover, multiple properties can match to one control (e.g., use of a combobox) as well as multiple controls matching to one property (e.g., two controls display the same information.)

[0072] Act **1012** is integrating the control with a graphical user interface (GUI.) A GUI allows a user to interact with a control and the control can be matched with at least one property. For example, the control can bind with several different properties through execution of a combobox. An example integration allows an arrow to be placed on the control that allows a user to scroll through different properties by engaging the arrow. Presenting the graphical user interface **1014** occurs which allows a user to view the GUI and to engage the GUI.

[0073] FIG. **11** discloses an automatic binding construction methodology **1100** with a master-detail implementation. There is generating a model with at least one property **1102**. A model can be a representation of a database including tables that populate the database as well as different relationships between the tables. Generation can take place automatically though use of mapping files with a database.

[0074] A check **1104** takes place if a master-detail configuration is appropriate. If a master-detail configuration is appropriate, then there is enabling a master-detail implementation **1106**. A master-detail implementation allows for further binding to show relationships between different properties in a model. In an example master-detail implementation, a form holding a control displays that a table 'people' exists, but there is also a desire to match 'friends' automatically that relate to the 'people'. 'People' can be the master record while 'friends' are considered detail. If a master-detail implementation is not appropriate, then there is enabling of a default implementation **1108**. Various default implementations (e.g., parent-child implementation, etc.) are available to be used in creating a binding automatically at runtime.

[0075] Analyzing at least one control on a form **1110** occurs. Analysis can include a number of different examinations. One common examination is determining if a control is capable of receiving a proper bind. For example, a control can have a relatively small display area while information that is intended to be bound is relatively large. Since this control could a poor choice for binding with intended information, this can be beneficial information. According to one embodiment, analyzing a control on a form is locating the control. Other actions that are non-exhaustive examples of analysis are determining if a control is already bound, investigating a form to determine what controls are on the form, performing a virus check on a control, etc.

[0076] There is matching a least one control with at least one property **1112**. Matching a control with a property creates a binding between them and this can take place automatically at runtime. Matching can include various verification actions. For instance, a control can already have a binding (e.g., be configured to display another property of another application.) Event **1112** can make this determination and match the property with another control. Moreover, multiple properties can match to one control (e.g., use of a combobox) as well as multiple controls matching to one property (e.g., two controls display the same information.)

[0077] Act **1114** is integrating the control with a graphical user interface (GUI.) A GUI allows a user to interact with a control and the control can be matched with at least one property. For example, the control can bind with several different properties through execution of a combobox. An example integration allows an arrow to be placed on the

control that allows a user to scroll through different properties by engaging the arrow. Presenting the graphical user interface can also occur.

[0078] There is creating a record related to matching at least one control with at least one property **1116**. Records of matches can be beneficial to further operations of the methodology **1100**. For instance, if a particular control has difficulties accepting a match, then in other engagements of the methodology **1100**, a larger amount of resources can be dedicated to binding. Moreover, since matches derive from computer code written by a user, an inference can be made (e.g., though artificial intelligence) that a programmer prefers certain binding types. A record of preferred binding types can be saved in a profile and user in later instances of the methodology **1100**.

[0079] FIG. **12** discloses a methodology **1200** for verifying automatic binding creation at runtime. There is receiving notification of a start of an application **1202**. Since operation of at least one action of the methodology **1200** is designed to take place during runtime, then it can become beneficial to know when runtime begins. According to one embodiment, an application transfers a message that is received at action **1202** to another location.

[0080] Estimating likelihood of at least one control type being proper **1204** occurs. While code written by a developer is intended to be less error prone (e.g., writing two lines of code as opposed to conventionally writing twenty lines of code), there is still a possibility of an error occurring. For instance, a user can write code that multiple properties link with a listbox. This can cause an error in some systems that operate the methodology **1200**. It is possible that event **1204** integrate with a cancellation ability where is a likelihood is too low, then matching is cancelled **1206**.

[0081] Action **1208** is analyzing at least on control on a form. Analysis can include a number of different examinations. One common examination is determining if a control is capable of receiving a proper bind. For example, a control can have a relatively small display area while information that is intended to be bound is relatively large. Since this control could a poor choice for binding with intended information, this can be beneficial information. According to one embodiment, analyzing a control on a form is locating the control. Other actions that are non-exhaustive examples of analysis are determining if a control is already bound, investigating a form to determine what controls are on the form, performing a virus check on a control, etc.

[0082] A check can take place to determine if a control type is available **1210**. For instance, a developer can make a request in code that checkbox be used as a control. However, a system operating the methodology **1200** could not have a checkbox control in a repertoire. Therefore, if the control type is not available, the matching is canceled **1206**. However, other possibilities are can be implemented, such as accessing a modification list that can attempt to solve the error (e.g., converting the checkbox selected to a textbox automatically.)

[0083] There is matching at least one control with at least one property **1212**. Matching a control with a property creates a binding between them and this can take place automatically at runtime. Matching can include various verification actions. For instance, a control can already have a binding (e.g., be configured to display another property of another application.) Event **1212** can make this determination and match the property with another control. Moreover, multiple properties can match to one control (e.g., use of a combobox) as well as multiple controls matching to one property (e.g., two controls display the same information.)

[0084] A check occurs to determine if a match is acceptable **1214**. A match can take place that contains an error. For instance, a control can contain a character limit display (e.g., display limited to eight characters) that does not disclose a person's name as requested (e.g., Larry_Thomas.) If a match is not acceptable, then the methodology **1200** can return to action **1212** and another matching can be attempted. A device operating the methodology **1200** can learn from an unacceptable match and make modification in future engagements (e.g., active learning through artificial intelligence.) Event **1216** is transferring a message that the match is complete and/or acceptable. Various units can desire to know that a link is successfully completed. For instance, a display can wait until a binding is complete before presenting a form.

[0085] In order to provide a context for the various aspects of the disclosed subject matter, FIGS. **13** and **14** as well as the following discussion are intended to provide a brief, general description of a suitable environment in which the various aspects of the disclosed subject matter can be implemented. While the subject matter has been described above in the general context of computer-executable instructions of a program that runs on one or more computers, those skilled in the art will recognize that the subject matter described herein also can be implemented in combination with other program modules. Generally, program modules include routines, programs, components, data structures, etc. that perform particular tasks and/or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the inventive methods can be practiced with other computer system configurations, including single-processor, multiprocessor or multi-core processor computer systems, mini-computing devices, mainframe computers, as well as personal computers, hand-held computing devices (e.g., personal digital assistant (PDA), phone, watch . . .), microprocessor-based or programmable consumer or industrial electronics, and the like. The illustrated aspects can also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. However, some, if not all aspects of the claimed subject matter can be practiced on stand-alone computers. In a distributed computing environment, program modules can be located in both local and remote memory storage devices.

[0086] Referring now to FIG. **13**, there is illustrated a schematic block diagram of a computing environment **1300** in accordance with the subject specification. The system **1300** includes one or more client(s) **1302**. The client(s) **1302** can be hardware and/or software (e.g., threads, processes, computing devices). The client(s) **1302** can house cookie(s) and/or associated contextual information by employing the specification, for example.

[0087] The system **1300** also includes one or more server(s) **1304**. The server(s) **1304** can also be hardware and/or software (e.g., threads, processes, computing devices). The servers **1304** can house threads to perform transformations by employing the specification, for example. One possible communication between a client **1302** and a server **1304** can be in the form of a data packet adapted to be transmitted between two or more computer processes. The data packet may include a cookie and/or associated contextual information, for example. The system **1300** includes a communication framework **1306** (e.g., a global communication network such

as the Internet) that can be employed to facilitate communications between the client(s) **1302** and the server(s) **1304**.

[0088] Communications can be facilitated via a wired (including optical fiber) and/or wireless technology. The client(s) **1302** are operatively connected to one or more client data store(s) **1308** that can be employed to store information local to the client(s) **1302** (e.g., cookie(s) and/or associated contextual information). Similarly, the server(s) **1304** are operatively connected to one or more server data store(s) **1310** that can be employed to store information local to the servers **1304**.

[0089] Referring now to FIG. **14**, there is illustrated a block diagram of a computer operable to execute the disclosed architecture. In order to provide additional context for various aspects of the subject specification, FIG. **14** and the following discussion are intended to provide a brief, general description of a suitable computing environment **1400** in which the various aspects of the specification can be implemented. While the specification has been described above in the general context of computer-executable instructions that may run on one or more computers, those skilled in the art will recognize that the specification also can be implemented in combination with other program modules and/or as a combination of hardware and software.

[0090] Generally, program modules include routines, programs, components, data structures, etc., that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the inventive methods can be practiced with other computer system configurations, including single-processor or multiprocessor computer systems, minicomputers, mainframe computers, as well as personal computers, hand-held computing devices, microprocessor-based or programmable consumer electronics, and the like, each of which can be operatively coupled to one or more associated devices.

[0091] The illustrated aspects of the specification may also be practiced in distributed computing environments where certain tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules can be located in both local and remote memory storage devices.

[0092] A computer typically includes a variety of computer-readable media. Computer-readable media can be any available media that can be accessed by the computer and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer-readable media can comprise computer storage media and communication media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disk (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the computer.

[0093] Communication media typically embodies computer-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism, and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer-readable media.

[0094] With reference again to FIG. **14**, the example environment **1400** for implementing various aspects of the specification includes a computer **1402**, the computer **1402** including a processing unit **1404**, a system memory **1406** and a system bus **1408**. The system bus **1408** couples system components including, but not limited to, the system memory **1406** to the processing unit **1404**. The processing unit **1404** can be any of various commercially available processors. Dual microprocessors and other multi-processor architectures may also be employed as the processing unit **1404**.

[0095] The system bus **1408** can be any of several types of bus structure that may further interconnect to a memory bus (with or without a memory controller), a peripheral bus, and a local bus using any of a variety of commercially available bus architectures. The system memory **1406** includes read-only memory (ROM) **1410** and random access memory (RAM) **1412**. A basic input/output system (BIOS) is stored in a non-volatile memory **1410** such as ROM, EPROM, EEPROM, which BIOS contains the basic routines that help to transfer information between elements within the computer **1402**, such as during start-up. The RAM **1412** can also include a high-speed RAM such as static RAM for caching data.

[0096] The computer **1402** further includes an internal hard disk drive (HDD) **1414** (e.g., EIDE, SATA), which internal hard disk drive **1414** may also be configured for external use in a suitable chassis (not shown), a magnetic floppy disk drive (FDD) **1416**, (e.g., to read from or write to a removable diskette **1418**) and an optical disk drive **1420**, (e.g., reading a CD-ROM disk **1422** or, to read from or write to other high capacity optical media such as the DVD). The hard disk drive **1414**, magnetic disk drive **1416** and optical disk drive **1420** can be connected to the system bus **1408** by a hard disk drive interface **1424**, a magnetic disk drive interface **1426** and an optical drive interface **1428**, respectively. The interface **1424** for external drive implementations includes at least one or both of Universal Serial Bus (USB) and IEEE 1394 interface technologies. Other external drive connection technologies are within contemplation of the subject specification.

[0097] The drives and their associated computer-readable media provide nonvolatile storage of data, data structures, computer-executable instructions, and so forth. For the computer **1402**, the drives and media accommodate the storage of any data in a suitable digital format. Although the description of computer-readable media above refers to a HDD, a removable magnetic diskette, and a removable optical media such as a CD or DVD, it should be appreciated by those skilled in the art that other types of media which are readable by a computer, such as zip drives, magnetic cassettes, flash memory cards, cartridges, and the like, may also be used in the example operating environment, and further, that any such media may contain computer-executable instructions for performing the methods of the specification.

[0098] A number of program modules can be stored in the drives and RAM **1412**, including an operating system **1430**, one or more application programs **1432**, other program mod-

ules **1434** and program data **1436**. All or portions of the operating system, applications, modules, and/or data can also be cached in the RAM **1412**. It is appreciated that the specification can be implemented with various commercially available operating systems or combinations of operating systems.

[0099] A user can enter commands and information into the computer **1402** through one or more wired/wireless input devices, e.g., a keyboard **1438** and a pointing device, such as a mouse **1440**. Other input devices (not shown) may include a microphone, an IR remote control, a joystick, a game pad, a stylus pen, touch screen, or the like. These and other input devices are often connected to the processing unit **1404** through an input device interface **1442** that is coupled to the system bus **1408**, but can be connected by other interfaces, such as a parallel port, an IEEE 1394 serial port, a game port, a USB port, an IR interface, etc.

[0100] A monitor **1444** or other type of display device is also connected to the system bus **1408** via an interface, such as a video adapter **1446**. In addition to the monitor **1444**, a computer typically includes other peripheral output devices (not shown), such as speakers, printers, etc.

[0101] The computer **1402** may operate in a networked environment using logical connections via wired and/or wireless communications to one or more remote computers, such as a remote computer(s) **1448**. The remote computer(s) **1448** can be a workstation, a server computer, a router, a personal computer, portable computer, microprocessor-based entertainment appliance, a peer device or other common network node, and typically includes many or all of the elements described relative to the computer **1402**, although, for purposes of brevity, only a memory/storage device **1450** is illustrated. The logical connections depicted include wired/wireless connectivity to a local area network (LAN) **1452** and/or larger networks, e.g., a wide area network (WAN) **1454**. Such LAN and WAN networking environments are commonplace in offices and companies, and facilitate enterprise-wide computer networks, such as intranets, all of which may connect to a global communications network, e.g., the Internet.

[0102] When used in a LAN networking environment, the computer **1402** is connected to the local network **1452** through a wired and/or wireless communication network interface or adapter **1456**. The adapter **1456** may facilitate wired or wireless communication to the LAN **1452**, which may also include a wireless access point disposed thereon for communicating with the wireless adapter **1456**.

[0103] When used in a WAN networking environment, the computer **1402** can include a modem **1458**, or is connected to a communications server on the WAN **1454**, or has other means for establishing communications over the WAN **1454**, such as by way of the Internet. The modem **1458**, which can be internal or external and a wired or wireless device, is connected to the system bus **1408** via the serial port interface **1442**. In a networked environment, program modules depicted relative to the computer **1402**, or portions thereof, can be stored in the remote memory/storage device **1450**. It will be appreciated that the network connections shown are example and other means of establishing a communications link between the computers can be used.

[0104] The computer **1402** is operable to communicate with any wireless devices or entities operatively disposed in wireless communication, e.g., a printer, scanner, desktop and/or portable computer, portable data assistant, communications satellite, any piece of equipment or location associated with a wirelessly detectable tag (e.g., a kiosk, news stand, restroom), and telephone. This includes at least Wi-Fi and Bluetooth™ wireless technologies. Thus, the communication can be a predefined structure as with a conventional network or simply an ad hoc communication between at least two devices.

[0105] Wi-Fi, or Wireless Fidelity, allows connection to the Internet from a couch at home, a bed in a hotel room, or a conference room at work, without wires. Wi-Fi is a wireless technology similar to that used in a cell phone that enables such devices, e.g., computers, to send and receive data indoors and out; anywhere within the range of a base station. Wi-Fi networks use radio technologies called IEEE 802.11(a, b, g, etc.) to provide secure, reliable, fast wireless connectivity. A Wi-Fi network can be used to connect computers to each other, to the Internet, and to wired networks (which use IEEE 802.3 or Ethernet). Wi-Fi networks operate in the unlicensed 2.4 and 5 GHz radio bands, at an 11 Mbps (802.11a) or 54 Mbps (802.11b) data rate, for example, or with products that contain both bands (dual band), so the networks can provide real-world performance similar to the basic 10 BaseT wired Ethernet networks used in many offices.

[0106] What has been described above includes examples of the subject specification. It is, of course, not possible to describe every conceivable combination of components or methodologies for purposes of describing the subject specification, but one of ordinary skill in the art may recognize that many further combinations and permutations of the subject specification are possible. Accordingly, the subject specification is intended to embrace all such alterations, modifications and variations that fall within the spirit and scope of the appended claims. Furthermore, to the extent that the term "includes" is used in either the detailed description or the claims, such term is intended to be inclusive in a manner similar to the term "comprising" as "comprising" is interpreted when employed as a transitional word in a claim.

What is claimed is:

1. A system, comprising:

a communication component that obtains data that relates to runtime operation of an application; and

a bind component that creates a link between a control and information based at least in part on the obtained data.

2. The system of claim **1**, wherein the bind component can use a name convention to create the link.

3. The system of claim **1**, wherein the bind component creates at least two links, wherein at least one link integrates with a first application and at least one link integrates with a second application.

4. The system of claim **1**, further comprising an add component that populates an index with at least one piece of data during runtime, wherein the communication component obtains at least one piece of index data.

5. The system of claim **1**, further comprising an authentication component that determines if operation of the bind component is proper.

6. The system of claim **1**, further comprising an analysis component that determines at least one asset related to a control.

7. The system of claim **1**, further comprising an aggregation component that builds information based on material from at least two sources, wherein the information is provided to the bind component for linkage.

8. The system of claim **1**, further comprising a search component that performs an investigation to locate data for obtainment by the communication component.

9. The system of claim **1**, wherein at least two details are disclosed, a first detail discloses at least one link that was created and a second detail discloses at least one link that was intended to be created, but was not created.

10. The system of claim **1**, further comprising an artificial intelligence component that makes at least one inference or at least one determination or at least one of each in relation to creation of a link between a control and information.

11. The system of claim **1**, further comprising a display component that presents at least one control linked with information through the bind component.

12. A method for automatic binding, comprising:

analyzing at least one control on a form; and

matching at least one control with at least one property.

13. The method of claim **12**, further comprising receiving a control name, a query, a property, or a combination thereof.

14. The method of claim **12**, further comprising indexing at least one query, at least one property, or at least one of each.

15. The method of claim **12**, further comprising determining a control type.

16. The method of claim **12**, further comprising integrating the control with a graphical user interface.

17. The method of claim **16**, further comprising presenting the graphical user interface.

18. A system for binding generation at runtime, comprising

means for analyzing a model; and

means for binding the model to at least one control.

19. The system of claim **18**, wherein the means for binding the model to at least one control utilizes a master-detail implementation.

20. The system of claim **18**, wherein the control integrates with a graphical user interface.

* * * * *