

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
10 May 2007 (10.05.2007)

PCT

(10) International Publication Number
WO 2007/051707 A2

(51) International Patent Classification:
G06F 9/44 (2006.01)

(21) International Application Number:
PCT/EP2006/067528

(22) International Filing Date: 18 October 2006 (18.10.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/266,698 3 November 2005 (03.11.2005) US

(71) Applicant (for all designated States except US): **INTERNATIONAL BUSINESS MACHINES CORPORATION** [US/US]; New Orchard Road, Armonk, New York 10504 (US).

(71) Applicant (for MG only): **IBM UNITED KINGDOM LIMITED** [GB/GB]; PO Box 41, Portsmouth Hampshire PO6 3AU (GB).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **BODIN, William,**

Kress [US/US]; 9400 Graceland Trail, Austin, Texas 78717 (US). **THORSON, Derral** [US/US]; 16829 Squaw Valley Lane, Austin, Texas 78717 (US). **REDMAN, Jerry, Wayne** [US/US]; 2705 Beechnut Trace, Cedar Park, Texas 78613 (US). **JARAMILLO, David** [US/US]; 6901 Finamore Circle, Lake Worth, Florida 33467 (US).

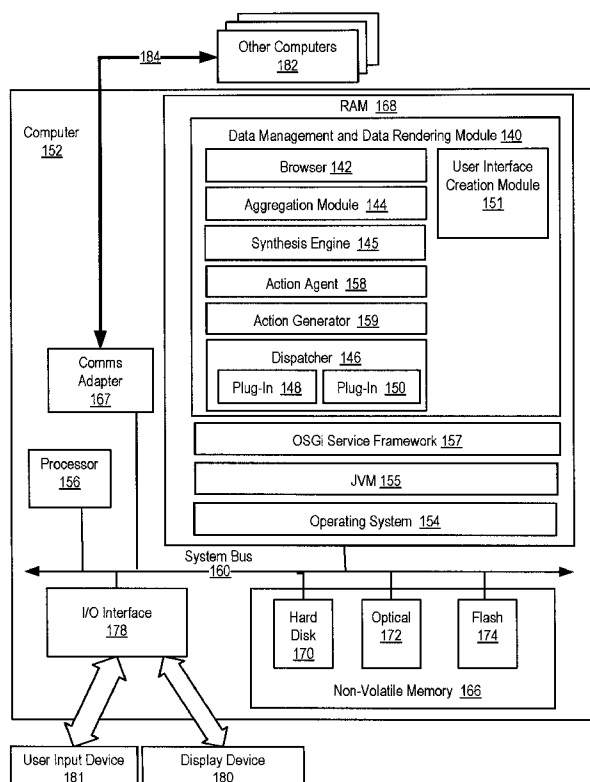
(74) Agent: **SEKAR, Anita**; IBM United Kingdom Limited, Intellectual Property Law, Winchester Hampshire SO21 2JN (GB).

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

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH,

[Continued on next page]

(54) Title: DYNAMIC CREATION OF USER INTERFACES FOR DATA MANAGEMENT AND DATA RENDERING



(57) Abstract: Methods, systems, and products are disclosed for dynamic creation of user interfaces for data management and data rendering including receiving an identification of a data source; retrieving data from the identified data source; and dynamically creating, in dependence upon the retrieved data, a user interface object to access the identified data source. The identified data source may be a web page on a server, in which case retrieving data from the identified data source may also include sending an HTTP request for the web page to a server and receiving an HTTP response containing the web page contents from the server. Dynamic creation of user interfaces for data management and data rendering may also include assigning the identified data source to a channel. Assigning the identified data source to a channel also include assigning the identified data source to a channel in dependence upon other synthesized content in the channel.



GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report*

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

**DYNAMIC CREATION OF USER INTERFACES FOR DATA
MANAGEMENT AND DATA RENDERING**

FIELD OF THE INVENTION

The field of the invention is data processing, or, more specifically, methods, systems, and products for dynamic creation of user interfaces for data management and data rendering.

BACKGROUND OF THE INVENTION

Despite having more access to data and having more devices to access that data, users are often time constrained. One reason for this time constraint is that users typically must access data of disparate data types from disparate data sources on data type-specific devices using data type-specific applications. One or more such data type-specific devices may be cumbersome for use at a particular time due to any number of external circumstances. Examples of external circumstances that may make data type-specific devices cumbersome to use include crowded locations, uncomfortable locations such as a train or car, user activity such as walking, visually intensive activities such as driving, and others as will occur to those of skill in the art. There is therefore an ongoing need for data management and data rendering for disparate data types that provides access to uniform data type access to content from disparate data sources.

DISCLOSURE OF THE INVENTION

Methods, systems, and products are disclosed for dynamic creation of user interfaces for data management and data rendering including receiving an identification of a data source; retrieving data from the identified data source; and dynamically creating, in dependence upon the retrieved data, a user interface object to access the identified data source. The identified data source may be a web page on a server, in which case retrieving data from the identified data source may also include sending an HTTP request for the web page to a server and receiving an HTTP response containing the web page contents from the server. Dynamic

creation of user interfaces for data management and data rendering may also include assigning the identified data source to a channel. Assigning the identified data source to a channel also include assigning the identified data source to a channel in dependence upon other synthesized content in the channel.

Dynamically creating, in dependence upon the retrieved data, a user interface object to access the identified data source may include creating display text describing source information. Dynamically creating, in dependence upon the retrieved data, a user interface object to access the identified data source may also include creating grammars to voice enable the user interface object.

The foregoing and other features and advantages of the invention will be apparent from the following more particular descriptions of exemplary embodiments of the invention as illustrated in the accompanying drawings wherein like reference numbers generally represent like parts of exemplary embodiments of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 sets forth a network diagram illustrating an exemplary system for data management and data rendering for disparate data types according to embodiments of the present invention.

Figure 2 sets forth a block diagram of automated computing machinery comprising an exemplary computer useful in data management and data rendering for disparate data types according to embodiments of the present invention.

Figure 3 sets forth a block diagram depicting a system for data management and data rendering for disparate data types according to of the present invention.

Figure 4 sets forth a flow chart illustrating an exemplary method for data management and data rendering for disparate data types according to embodiments of the present invention.

Figure 5 sets forth a flow chart illustrating an exemplary method for aggregating data of disparate data types from disparate data sources according to embodiments of the present invention.

5 Figure 6 sets forth a flow chart illustrating an exemplary method for retrieving, from the identified data source, the requested data according to embodiments of the present invention.

10 Figure 7 sets forth a flow chart illustrating an exemplary method for aggregating data of disparate data types from disparate data sources according to the present invention.

15 Figure 8 sets forth a flow chart illustrating an exemplary method for aggregating data of disparate data types from disparate data sources according to the present invention.

20 Figure 9 sets forth a flow chart illustrating a exemplary method for synthesizing aggregated data of disparate data types into data of a uniform data type according to the present invention.

 Figure 10 sets forth a flow chart illustrating a exemplary method for synthesizing aggregated data of disparate data types into data of a uniform data type according to the present invention.

25 Figure 11 sets forth a flow chart illustrating an exemplary method for identifying an action in dependence upon the synthesized data according to the present invention.

30 Figure 12 sets forth a flow chart illustrating an exemplary method for channelizing the synthesized data according to embodiments of the present invention.

35 Figure 13 sets forth a flow chart illustrating an exemplary method for dynamic creation of user interfaces for data management and data rendering according to embodiments of the present invention.

 Figure 14 sets forth a line drawing of an exemplary web page displaying available data sources and a configuration page designed to receive identifications of data sources and present display text

describing source information of the identified data source in accordance with the present invention.

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

Exemplary methods, systems, and products for data management and data rendering for disparate data types from disparate data sources according to embodiments of the present invention are described with reference to the accompanying drawings, beginning with Figure 1. Figure 1 sets forth a network diagram illustrating an exemplary system for data management and data rendering for disparate data types according to embodiments of the present invention. The system of Figure 1 operates generally to manage and render data for disparate data types according to embodiments of the present invention by aggregating data of disparate data types from disparate data sources, synthesizing the aggregated data of disparate data types into data of a uniform data type, identifying an action in dependence upon the synthesized data, and executing the identified action.

Disparate data types are data of different kind and form. That is, disparate data types are data of different kinds. The distinctions in data that define the disparate data types may include a difference in data structure, file format, protocol in which the data is transmitted, and other distinctions as will occur to those of skill in the art. Examples of disparate data types include MPEG-1 Audio Layer 3 ('MP3') files, Extensible markup language documents ('XML'), email documents, and so on as will occur to those of skill in the art. Disparate data types typically must be rendered on data type-specific devices. For example, an MPEG-1 Audio Layer 3 ('MP3') file is typically played by an MP3 player, a Wireless Markup Language ('WML') file is typically accessed by a wireless device, and so on.

The term disparate data sources means sources of data of disparate data types. Such data sources may be any device or network location capable of providing access to data of a disparate data type. Examples of disparate data sources include servers serving up files, web sites, cellular phones, PDAs, MP3 players, and so on as will occur to those of skill in the art.

The system of Figure 1 includes a number of devices operating as disparate data sources connected for data communications in networks. The data processing system of Figure 1 includes a wide area network ("WAN") (110) and a local area network ("LAN") (120). "LAN" is an abbreviation for "local area network." A LAN is a computer network that spans a relatively small area. Many LANs are confined to a single building or group of buildings. However, one LAN can be connected to other LANs over any distance via telephone lines and radio waves. A system of LANs connected in this way is called a wide-area network (WAN). The Internet is an example of a WAN.

In the example of Figure 1, server (122) operates as a gateway between the LAN (120) and the WAN (110). The network connection aspect of the architecture of Figure 1 is only for explanation, not for limitation. In fact, systems for data management and data rendering for disparate data types according to embodiments of the present invention may be connected as LANs, WANs, intranets, internets, the Internet, webs, the World Wide Web itself, or other connections as will occur to those of skill in the art. Such networks are media that may be used to provide data communications connections between various devices and computers connected together within an overall data processing system.

In the example of Figure 1, a plurality of devices are connected to a LAN and WAN respectively, each implementing a data source and each having stored upon it data of a particular data type. In the example of Figure 1, a server (108) is connected to the WAN through a wireline connection (126). The server (108) of Figure 1 is a data source for an RSS feed, which the server delivers in the form of an XML file. RSS is a family of XML file formats for web syndication used by news websites and weblogs. The abbreviation is used to refer to the following standards: Rich Site Summary (RSS 0.91), RDF Site Summary (RSS 0.9, 1.0 and 1.1), and Really Simple Syndication (RSS 2.0). The RSS formats provide web content or summaries of web content together with links to the full versions of the content, and other meta-data. This information is delivered as an XML file called RSS feed, webfeed, RSS stream, or RSS channel.

In the example of Figure 1, another server (106) is connected to the WAN through a wireline connection (132). The server (106) of Figure 1 is a data source for data stored as a Lotus NOTES file. In the example of

Figure 1, a personal digital assistant ('PDA') (102) is connected to the WAN through a wireless connection (130). The PDA is a data source for data stored in the form of an XHTML Mobile Profile ('XHTML MP') document.

5 In the example of Figure 1, a cellular phone (104) is connected to the WAN through a wireless connection (128). The cellular phone is a data source for data stored as a Wireless Markup Language ('WML') file. In the example of Figure 1, a tablet computer (112) is connected to the WAN through a wireless connection (134). The tablet computer (112) is a data
10 source for data stored in the form of an XHTML MP document.

The system of Figure 1 also includes a digital audio player ('DAP') (116). The DAP (116) is connected to the LAN through a wireline connection (192). The digital audio player ('DAP') (116) of Figure 1 is a
15 data source for data stored as an MP3 file. The system of Figure 1 also includes a laptop computer (124). The laptop computer is connected to the LAN through a wireline connection (190). The laptop computer (124) of Figure 1 is a data source data stored as a Graphics Interchange Format ('GIF') file. The laptop computer (124) of Figure 1 is also a data source
20 for data in the form of Extensible Hypertext Markup Language ('XHTML') documents.

The system of Figure 1 includes a laptop computer (114) and a smart phone (118) each having installed upon it a data management and rendering
25 module proving uniform access to the data of disparate data types available from the disparate data sources. The exemplary laptop computer (114) of Figure 1 connects to the LAN through a wireless connection (188). The exemplary smart phone (118) of Figure 1 also connects to the LAN through a wireless connection (186). The laptop computer (114) and smart
30 phone (118) of Figure 1 have installed and running on them software capable generally of data management and data rendering for disparate data types by aggregating data of disparate data types from disparate data sources; synthesizing the aggregated data of disparate data types into data of a uniform data type; identifying an action in dependence upon the
35 synthesized data; and executing the identified action.

Aggregated data is the accumulation, in a single location, of data of disparate types. This location of the aggregated data may be either physical, such as, for example, on a single computer containing aggregated

data, or logical, such as, for example, a single interface providing access to the aggregated data.

Synthesized data is aggregated data which has been synthesized into data of a uniform data type. The uniform data type may be implemented as text content and markup which has been translated from the aggregated data. Synthesized data may also contain additional voice markup inserted into the text content, which adds additional voice capability.

Alternatively, any of the devices of the system of Figure 1 described as sources may also support a data management and rendering module according to the present invention. For example, the server (106), as described above, is capable of supporting a data management and rendering module providing uniform access to the data of disparate data types available from the disparate data sources. Any of the devices of Figure 1, as described above, such as, for example, a PDA, a tablet computer, a cellular phone, or any other device as will occur to those of skill in the art, are capable of supporting a data management and rendering module according to the present invention.

The arrangement of servers and other devices making up the exemplary system illustrated in Figure 1 are for explanation, not for limitation. Data processing systems useful according to various embodiments of the present invention may include additional servers, routers, other devices, and peer-to-peer architectures, not shown in Figure 1, as will occur to those of skill in the art. Networks in such data processing systems may support many data communications protocols, including for example TCP (Transmission Control Protocol), IP (Internet Protocol), HTTP (HyperText Transfer Protocol), WAP (Wireless Access Protocol), HDTP (Handheld Device Transport Protocol), and others as will occur to those of skill in the art. Various embodiments of the present invention may be implemented on a variety of hardware platforms in addition to those illustrated in Figure 1.

A method for data management and data rendering for disparate data types in accordance with the present invention is generally implemented with computers, that is, with automated computing machinery. In the system of Figure 1, for example, all the nodes, servers, and communications devices are implemented to some extent at least as

computers. For further explanation, therefore, Figure 2 sets forth a block diagram of automated computing machinery comprising an exemplary computer (152) useful in data management and data rendering for disparate data types according to embodiments of the present invention. The computer (152) of Figure 2 includes at least one computer processor (156) or 'CPU' as well as random access memory (168) ('RAM') which is connected through a system bus (160) to a processor (156) and to other components of the computer.

Stored in RAM (168) is a data management and data rendering module (140), computer program instructions for data management and data rendering for disparate data types capable generally of aggregating data of disparate data types from disparate data sources; synthesizing the aggregated data of disparate data types into data of a uniform data type; identifying an action in dependence upon the synthesized data; and executing the identified action. Data management and data rendering for disparate data types advantageously provides to the user the capability to efficiently access and manipulate data gathered from disparate data type-specific resources. Data management and data rendering for disparate data types also provides a uniform data type such that a user may access data gathered from disparate data type-specific resources on a single device.

Also stored in RAM (168) is an aggregation module (144), computer program instructions for aggregating data of disparate data types from disparate data sources capable generally of receiving, from an aggregation process, a request for data; identifying, in response to the request for data, one of two or more disparate data sources as a source for data; retrieving, from the identified data source, the requested data; and returning to the aggregation process the requested data. Aggregating data of disparate data types from disparate data sources advantageously provides the capability to collect data from multiple sources for synthesis.

Also stored in RAM is a synthesis engine (145), computer program instructions for synthesizing aggregated data of disparate data types into data of a uniform data type capable generally of receiving aggregated data of disparate data types and translating each of the aggregated data of disparate data types into translated data composed of text content and

markup associated with the text content. Synthesizing aggregated data of disparate data types into data of a uniform data type advantageously provides synthesized data of a uniform data type which is capable of being accessed and manipulated by a single device.

5

Also stored in RAM (168) is an action generator module (159), a set of computer program instructions for identifying actions in dependence upon synthesized data and often user instructions. Identifying an action in dependence upon the synthesized data advantageously provides the capability of interacting with and managing synthesized data.

10

Also stored in RAM (168) is an action agent (158), a set of computer program instructions for administering the execution of one or more identified actions. Such execution may be executed immediately upon identification, periodically after identification, or scheduled after identification as will occur to those of skill in the art.

15

Also stored in RAM (168) is a dispatcher (146), computer program instructions for receiving, from an aggregation process, a request for data; identifying, in response to the request for data, one of a plurality of disparate data sources as a source for the data; retrieving, from the identified data source, the requested data; and returning, to the aggregation process, the requested data. Receiving, from an aggregation process, a request for data; identifying, in response to the request for data, one of a plurality of disparate data sources as a source for the data; retrieving, from the identified data source, the requested data; and returning, to the aggregation process, the requested data advantageously provides the capability to access disparate data sources for aggregation and synthesis.

20

25

30

The dispatcher (146) of Figure 2 also includes a plurality of plug-in modules (148, 150), computer program instructions for retrieving, from a data source associated with the plug-in, requested data for use by an aggregation process. Such plug-ins isolate the general actions of the dispatcher from the specific requirements needed to retrieved data of a particular type.

35

Also stored in RAM (168) is a browser (142), computer program instructions for providing an interface for the user to synthesized data.

Providing an interface for the user to synthesized data advantageously provides a user access to content of data retrieved from disparate data sources without having to use data source-specific devices. The browser (142) of Figure 2 is capable of multimodal interaction capable of receiving multimodal input and interacting with users through multimodal output. Such multimodal browsers typically support multimodal web pages that provide multimodal interaction through hierarchical menus that may be speech driven.

Also stored in RAM is a user interface creation module (151), computer program instruction capable generally of receiving an identification of a data source; retrieving data from the identified data source; and dynamically creating, in dependence upon the retrieved data, a user interface object to access the identified data source.

Also stored in RAM is an OSGi Service Framework (157) running on a Java Virtual Machine ('JVM') (155) (Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both). "OSGi" refers to the Open Service Gateway initiative, an industry organization developing specifications delivery of service bundles, software middleware providing compliant data communications and services through services gateways. The OSGi specification is a Java based application layer framework that gives service providers, network operator device makers, and appliance manufacturer's vendor neutral application and device layer APIs and functions. OSGi works with a variety of networking technologies like Ethernet, Bluetooth, the 'Home, Audio and Video Interoperability standard' (HAVi), IEEE 1394, Universal Serial Bus (USB), WAP, X-10, Lon Works, HomePlug and various other networking technologies. The OSGi specification is available for free download from the OSGi website at www.osgi.org.

An OSGi service framework (157) is written in Java and therefore, typically runs on a Java Virtual Machine (JVM) (155). In OSGi, the service framework (157) is a hosting platform for running 'services'. The term 'service' or 'services' in this disclosure, depending on context, generally refers to OSGi-compliant services.

Services are the main building blocks for creating applications according to the OSGi. A service is a group of Java classes and interfaces that implement a certain feature. The OSGi specification provides a number of standard services. For example, OSGi provides a standard HTTP service that creates a web server that can respond to requests from HTTP clients.

OSGi also provides a set of standard services called the Device Access Specification. The Device Access Specification ("DAS") provides services to identify a device connected to the services gateway, search for a driver for that device, and install the driver for the device.

Services in OSGi are packaged in 'bundles' with other files, images, and resources that the services need for execution. A bundle is a Java archive or 'JAR' file including one or more service implementations, an activator class, and a manifest file. An activator class is a Java class that the service framework uses to start and stop a bundle. A manifest file is a standard text file that describes the contents of the bundle.

The service framework (157) in OSGi also includes a service registry. The service registry includes a service registration including the service's name and an instance of a class that implements the service for each bundle installed on the framework and registered with the service registry. A bundle may request services that are not included in the bundle, but are registered on the framework service registry. To find a service, a bundle performs a query on the framework's service registry.

Data management and data rendering according to embodiments of the present invention may be usefully invoke one or more OSGi services. OSGi is included for explanation and not for limitation. In fact, data management and data rendering according embodiments of the present invention may usefully employ many different technologies and all such technologies are well within the scope of the present invention.

Also stored in RAM (168) is an operating system (154). Operating systems useful in computers according to embodiments of the present invention include UNIX, Linux, Microsoft Windows NT, AIX, IBM's i5/OS (UNIX is a registered trademark of The Open Group in the United States and other countries; Linux is a registered trademark of Linus Torvalds in the

United States, other countries, or both; Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both; AIX, IBM and i5/OS are registered trademarks of International Business Machines Corporation), and others as will occur to those of skill in the art. The operating system (154) and data management and data rendering module (140) in the example of Figure 2 are shown in RAM (168), but many components of such software typically are stored in non-volatile memory (166) also.

Computer (152) of Figure 2 includes non-volatile computer memory (166) coupled through a system bus (160) to a processor (156) and to other components of the computer (152). Non-volatile computer memory (166) may be implemented as a hard disk drive (170), an optical disk drive (172), an electrically erasable programmable read-only memory space (so-called 'EEPROM' or 'Flash' memory) (174), RAM drives (not shown), or as any other kind of computer memory as will occur to those of skill in the art.

The example computer of Figure 2 includes one or more input/output interface adapters (178). Input/output interface adapters in computers implement user-oriented input/output through, for example, software drivers and computer hardware for controlling output to display devices (180) such as computer display screens, as well as user input from user input devices (181) such as keyboards and mice.

The exemplary computer (152) of Figure 2 includes a communications adapter (167) for implementing data communications (184) with other computers (182). Such data communications may be carried out serially through RS-232 connections, through external buses such as a USB, through data communications networks such as IP networks, and in other ways as will occur to those of skill in the art. Communications adapters implement the hardware level of data communications through which one computer sends data communications to another computer, directly or through a network. Examples of communications adapters useful for data management and data rendering for disparate data types from disparate data sources according to embodiments of the present invention include modems for wired dial-up communications, Ethernet (IEEE 802.3) adapters for wired network communications, and 802.11b adapters for wireless network communications.

For further explanation, Figure 3 sets forth a block diagram depicting a system for data management and data rendering for disparate data types according to of the present invention. The system of Figure 3 includes an aggregation module (144), computer program instructions for aggregating data of disparate data types from disparate data sources capable generally of receiving, from an aggregation process, a request for data; identifying, in response to the request for data, one of two or more disparate data sources as a source for data; retrieving, from the identified data source, the requested data; and returning to the aggregation process the requested data.

The system of Figure 3 includes a synthesis engine (145), computer program instructions for synthesizing aggregated data of disparate data types into data of a uniform data type capable generally of receiving aggregated data of disparate data types and translating each of the aggregated data of disparate data types into translated data composed of text content and markup associated with the text content.

The synthesis engine (145) includes a VXML Builder (222) module, computer program instructions for translating each of the aggregated data of disparate data types into text content and markup associated with the text content. The synthesis engine (145) also includes a grammar builder (224) module, computer program instructions for generating grammars for voice markup associated with the text content.

The system of Figure 3 includes a synthesized data repository (226) data storage for the synthesized data created by the synthesis engine in X + V format. The system of Figure 3 also includes an X + V browser (142), computer program instructions capable generally of presenting the synthesized data from the synthesized data repository (226) to the user. Presenting the synthesized data may include both graphical display and audio representation of the synthesized data. As discussed below with reference to Figure 4, one way presenting the synthesized data to a user may be carried out is by presenting synthesized data through one or more channels.

The system of Figure 3 includes a dispatcher (146) module, computer program instructions for receiving, from an aggregation process, a request for data; identifying, in response to the request for data, one of a

plurality of disparate data sources as a source for the data; retrieving, from the identified data source, the requested data; and returning, to the aggregation process, the requested data. The dispatcher (146) module accesses data of disparate data types from disparate data sources for the aggregation module (144), the synthesis engine (145), and the action agent (158). The system of Figure 3 includes data source-specific plug-ins (148-150, 234-236) used by the dispatcher to access data as discussed below.

In the system of Figure 3, the data sources include local data (216) and content servers (202). Local data (216) is data contained in memory or registers of the automated computing machinery. In the system of Figure 3, the data sources also include content servers (202). The content servers (202) are connected to the dispatcher (146) module through a network (501). An RSS server (108) of Figure 3 is a data source for an RSS feed, which the server delivers in the form of an XML file. RSS is a family of XML file formats for web syndication used by news websites and weblogs. The abbreviation is used to refer to the following standards: Rich Site Summary (RSS 0.91), RDF Site Summary (RSS 0.9, 1.0 and 1.1), and Really Simple Syndication (RSS 2.0). The RSS formats provide web content or summaries of web content together with links to the full versions of the content, and other meta-data. This information is delivered as an XML file called RSS feed, webfeed, RSS stream, or RSS channel.

In the system of Figure 3, an email server (106) is a data source for email. The server delivers this email in the form of a Lotus NOTES file (Lotus Notes is a registered trademark of International Business Machines Corporation). In the system of Figure 3, a calendar server (107) is a data source for calendar information. Calendar information includes calendared events and other related information. The server delivers this calendar information in the form of a Lotus NOTES file.

In the system of Figure 3, an IBM On Demand Workstation (204) a server providing support for an On Demand Workplace ('ODW') that provides productivity tools, and a virtual space to share ideas and expertise, collaborate with others, and find information.

The system of Figure 3 includes data source-specific plug-ins (148-150, 234-236). For each data source listed above, the dispatcher uses a specific plug-in to access data.

5 The system of Figure 3 includes an RSS plug-in (148) associated with an RSS server (108) running an RSS application. The RSS plug-in (148) of Figure 3 retrieves the RSS feed from the RSS server (108) for the user and provides the RSS feed in an XML file to the aggregation module.

10 The system of Figure 3 includes a calendar plug-in (150) associated with a calendar server (107) running a calendaring application. The calendar plug-in (150) of Figure 3 retrieves calendared events from the calendar server (107) for the user and provides the calendared events to the aggregation module.

15 The system of Figure 3 includes an email plug-in (234) associated with an email server (106) running an email application. The email plug-in (234) of Figure 3 retrieves email from the email server (106) for the user and provides the email to the aggregation module.

20 The system of Figure 3 includes an On Demand Workstation ('ODW') plug-in (236) associated with an ODW server (204) running an ODW application. The ODW plug-in (236) of Figure 3 retrieves ODW data from the ODW server (204) for the user and provides the ODW data to the aggregation module.

25 The system of Figure 3 also includes an action generator module (159), computer program instructions for identifying an action from the action repository (240) in dependence upon the synthesized data capable generally of receiving a user instruction, selecting synthesized data in response to the user instruction, and selecting an action in dependence upon the user instruction and the selected data.

30 The action generator module (159) contains an embedded server (244). The embedded server (244) receives user instructions through the X + V browser (142). Upon identifying an action from the action repository (240), the action generator module (159) employs the action agent (158) to execute the action. The system of Figure 3 includes an action agent (158), computer program instructions for executing an action capable generally of executing actions.

For further explanation, Figure 4 sets forth a flow chart illustrating an exemplary method for data management and data rendering for disparate data types according to embodiments of the present invention. The method of Figure 4 includes aggregating (406) data of
5 disparate data types (402, 408) from disparate data sources (404, 410). As discussed above, aggregated data of disparate data types is the accumulation, in a single location, of data of disparate types. This location of the aggregated data may be either physical, such as, for example, on a single computer containing aggregated data, or logical, such
10 as, for example, a single interface providing access to the aggregated data.

Aggregating (406) data of disparate data types (402, 408) from disparate data sources (404, 410) according to the method of Figure 4 may
15 be carried out by receiving, from an aggregation process, a request for data; identifying, in response to the request for data, one of two or more disparate data sources as a source for data; retrieving, from the identified data source, the requested data; and returning to the aggregation process the requested data as discussed in more detail below
20 with reference to Figure 5.

The method of Figure 4 also includes synthesizing (414) the aggregated data of disparate data types (412) into data of a uniform data type. Data of a uniform data type is data having been created or
25 translated into a format of predetermined type. That is, uniform data types are data of a single kind that may be rendered on a device capable of rendering data of the uniform data type. Synthesizing (414) the aggregated data of disparate data types (412) into data of a uniform data type advantageously results in a single point of access for the content of
30 the aggregation of disparate data retrieved from disparate data sources.

One example of a uniform data type useful in synthesizing (414) aggregated data of disparate data types (412) into data of a uniform data type is XHTML plus Voice. XHTML plus Voice ('X+V') is a Web markup
35 language for developing multimodal applications, by enabling voice in a presentation layer with voice markup. X + V provides voice-based interaction in small and mobile devices using both voice and visual elements. X+V is composed of three main standards: XHTML, VoiceXML, and XML Events. Given that the Web application environment is event-driven,

X+V incorporates the Document Object Model (DOM) eventing framework used in the XML Events standard. Using this framework, X+V defines the familiar event types from HTML to create the correlation between visual and voice markup.

5
Synthesizing (414) the aggregated data of disparate data types (412) into data of a uniform data type may be carried out by receiving aggregated data of disparate data types and translating each of the aggregated data of disparate data types into text content and markup
10 associated with the text content as discussed in more detail with reference to Figure 9. In the method of Figure 4, synthesizing the aggregated data of disparate data types (412) into data of a uniform data type may be carried out by translating the aggregated data into X+V, or any other markup language as will occur to those of skill in the art.

15
The method for data management and data rendering of Figure 4 also includes identifying (418) an action in dependence upon the synthesized data (416). An action is a set of computer instructions that when executed carry out a predefined task. The action may be executed in
20 dependence upon the synthesized data immediately or at some defined later time. Identifying (418) an action in dependence upon the synthesized data (416) may be carried out by receiving a user instruction, selecting synthesized data in response to the user instruction, and selecting an action in dependence upon the user instruction and the selected data.

25
A user instruction is an event received in response to an act by a user. Exemplary user instructions include receiving events as a result of a user entering a combination of keystrokes using a keyboard or keypad, receiving speech from a user, receiving an event as a result of clicking
30 on icons on a visual display by using a mouse, receiving an event as a result of a user pressing an icon on a touchpad, or other user instructions as will occur to those of skill in the art. Receiving a user instruction may be carried out by receiving speech from a user, converting the speech to text, and determining in dependence upon the text and a
35 grammar the user instruction. Alternatively, receiving a user instruction may be carried out by receiving speech from a user and determining the user instruction in dependence upon the speech and a grammar.

The method of Figure 4 also includes executing (424) the identified action (420). Executing (424) the identified action (420) may be carried

out by calling a member method in an action object identified in dependence upon the synthesized data, executing computer program instructions carrying out the identified action, as well as other ways of executing an identified action as will occur to those of skill in the art. Executing (424) the identified action (420) may also include determining the availability of a communications network required to carry out the action and executing the action only if the communications network is available and postponing executing the action if the communications network connection is not available. Postponing executing the action if the communications network connection is not available may include enqueueing identified actions into an action queue, storing the actions until a communications network is available, and then executing the identified actions. Another way that waiting to execute the identified action (420) may be carried out is by inserting an entry delineating the action into a container, and later processing the container. A container could be any data structure suitable for storing an entry delineating an action, such as, for example, an XML file.

Executing (424) the identified action (420) may include modifying the content of data of one of the disparate data sources. Consider for example, an action called deleteOldEmail() that when executed deletes not only synthesized data translated from email, but also deletes the original source email stored on an email server coupled for data communications with a data management and data rendering module operating according to the present invention.

The method of Figure 4 also includes channelizing (422) the synthesized data (416). A channel is a logical aggregation of data content for presentation to a user. Channelizing (422) the synthesized data (416) may be carried out by identifying attributes of the synthesized data, characterizing the attributes of the synthesized data, and assigning the data to a predetermined channel in dependence upon the characterized attributes and channel assignment rules. Channelizing the synthesized data advantageously provides a vehicle for presenting related content to a user. Examples of such channelized data may be a 'work channel' that provides a channel of work related content, an 'entertainment channel' that provides a channel of entertainment content and so on as will occur to those of skill in the art.

The method of Figure 4 may also include presenting (426) the synthesized data (416) to a user through one or more channels. One way presenting (426) the synthesized data (416) to a user through one or more channels may be carried out is by presenting summaries or headings of available channels. The content presented through those channels can be accessed via this presentation in order to access the synthesized data (416). Another way presenting (426) the synthesized data (416) to a user through one or more channels may be carried out by displaying or playing the synthesized data (416) contained in the channel. Text might be displayed visually, or it could be translated into a simulated voice and played for the user.

For further explanation, Figure 5 sets forth a flow chart illustrating an exemplary method for aggregating data of disparate data types from disparate data sources according to embodiments of the present invention. In the method of Figure 5, aggregating (406) data of disparate data types (402, 408) from disparate data sources (404, 522) includes receiving (506), from an aggregation process (502), a request for data (508). A request for data may be implemented as a message, from the aggregation process, to a dispatcher instructing the dispatcher to initiate retrieving the requested data and returning the requested data to the aggregation process.

In the method of Figure 5, aggregating (406) data of disparate data types (402, 408) from disparate data sources (404, 522) also includes identifying (510), in response to the request for data (508), one of a plurality of disparate data sources (404, 522) as a source for the data. Identifying (510), in response to the request for data (508), one of a plurality of disparate data sources (404, 522) as a source for the data may be carried in a number of ways. One way of identifying (510) one of a plurality of disparate data sources (404, 522) as a source for the data may be carried out by receiving, from a user, an identification of the disparate data source; and identifying, to the aggregation process, the disparate data source in dependence upon the identification as discussed in more detail below with reference to Figure 7.

Another way of identifying, to the aggregation process (502), disparate data sources is carried out by identifying, from the request for data, data type information and identifying from the data source table

sources of data that correspond to the data type as discussed in more detail below with reference to Figure 8. Still another way of identifying one of a plurality of data sources is carried out by identifying, from the request for data, data type information; searching, in dependence upon the data type information, for a data source; and identifying from the search results returned in the data source search, sources of data corresponding to the data type also discussed below in more detail with reference to Figure 8.

The three methods for identifying one of a plurality of data sources described in this specification are for explanation and not for limitation. In fact, there are many ways of identifying one of a plurality of data sources and all such ways are well within the scope of the present invention.

The method for aggregating (406) data of Figure 5 includes retrieving (512), from the identified data source (522), the requested data (514). Retrieving (512), from the identified data source (522), the requested data (514) includes determining whether the identified data source requires data access information to retrieve the requested data; retrieving, in dependence upon data elements contained in the request for data, the data access information if the identified data source requires data access information to retrieve the requested data; and presenting the data access information to the identified data source as discussed in more detail below with reference to Figure 6. Retrieving (512) the requested data according the method of Figure 5 may be carried out by retrieving the data from memory locally, downloading the data from a network location, or any other way of retrieving the requested data that will occur to those of skill in the art. As discussed above, retrieving (512), from the identified data source (522), the requested data (514) may be carried out by a data-source-specific plug-in designed to retrieve data from a particular data source or a particular type of data source.

In the method of Figure 5, aggregating (406) data of disparate data types (402, 408) from disparate data sources (404, 522) also includes returning (516), to the aggregation process (502), the requested data (514). Returning (516), to the aggregation process (502), the requested data (514) returning the requested data to the aggregation process in a message, storing the data locally and returning a pointer pointing to the

location of the stored data to the aggregation process, or any other way of returning the requested data that will occur to those of skill in the art.

5 As discussed above with reference to Figure 5, aggregating (406) data of Figure 5 includes retrieving, from the identified data source, the requested data. For further explanation, therefore, Figure 6 sets forth a flow chart illustrating an exemplary method for retrieving (512), from the identified data source (522), the requested data (514) according to
10 embodiments of the present invention. In the method of Figure 6, retrieving (512), from the identified data source (522), the requested data (514) includes determining (904) whether the identified data source (522) requires data access information (914) to retrieve the requested data (514). As discussed above in reference to Figure 5, data access
15 information is information which is required to access some types of data from some of the disparate sources of data. Exemplary data access information includes account names, account numbers, passwords, or any other data access information that will occur to those of skill in the art.

20 Determining (904) whether the identified data source (522) requires data access information (914) to retrieve the requested data (514) may be carried out by attempting to retrieve data from the identified data source and receiving from the data source a prompt for data access information
25 required to retrieve the data. Alternatively, instead of receiving a prompt from the data source each time data is retrieved from the data source, determining (904) whether the identified data source (522) requires data access information (914) to retrieve the requested data (514) may be carried out once by, for example a user, and provided to a
30 dispatcher such that the required data access information may be provided to a data source with any request for data without prompt. Such data access information may be stored in, for example, a data source table identifying any corresponding data access information needed to access data from the identified data source.

35 In the method of Figure 6, retrieving (512), from the identified data source (522), the requested data (514) also includes retrieving (912), in dependence upon data elements (910) contained in the request for data (508), the data access information (914), if the identified data

source requires data access information to retrieve the requested data (908). Data elements (910) contained in the request for data (508) are typically values of attributes of the request for data (508). Such values may include values identifying the type of data to be accessed, values identifying the location of the disparate data source for the requested data, or any other values of attributes of the request for data.

Such data elements (910) contained in the request for data (508) are useful in retrieving data access information required to retrieve data from the disparate data source. Data access information needed to access data sources for a user may be usefully stored in a record associated with the user indexed by the data elements found in all requests for data from the data source. Retrieving (912), in dependence upon data elements (910) contained in the request for data (508), the data access information (914) according to Figure 6 may therefore be carried out by retrieving, from a database in dependence upon one or more data elements in the request, a record containing the data access information and extracting from the record the data access information. Such data access information may be provided to the data source to retrieve the data.

Retrieving (912), in dependence upon data elements (910) contained in the request for data (508), the data access information (914), if the identified data source requires data access information (914) to retrieve the requested data (908), may be carried out by identifying data elements (910) contained in the request for data (508), parsing the data elements to identify data access information (914) needed to retrieve the requested data (908), identifying in a data access table the correct data access information, and retrieving the data access information (914).

The exemplary method of Figure 6 for retrieving (512), from the identified data source (522), the requested data (514) also includes presenting (916) the data access information (914) to the identified data source (522). Presenting (916) the data access information (914) to the identified data source (522) according to the method of Figure 6 may be carried out by providing in the request the data access information as parameters to the request or providing the data access information in response to a prompt for such data access information by a data source. That is, presenting (916) the data access information (914) to the identified data source (522) may be carried out by a selected data source

specific plug-in of a dispatcher that provides data access information (914) for the identified data source (522) in response to a prompt for such data access information. Alternatively, presenting (916) the data access information (914) to the identified data source (522) may be carried out by a selected data source specific plug-in of a dispatcher that passes as parameters to request the data access information (914) for the identified data source (522) without prompt.

As discussed above, aggregating data of disparate data types from disparate data sources according to embodiments of the present invention typically includes identifying, to the aggregation process, disparate data sources. That is, prior to requesting data from a particular data source, that data source typically is identified to an aggregation process. For further explanation, therefore, Figure 7 sets forth a flow chart illustrating an exemplary method for aggregating data of disparate data types (404, 522) from disparate data sources (404, 522) according to the present invention that includes identifying (1006), to the aggregation process (502), disparate data sources (1008). In the method of Figure 7, identifying (1006), to the aggregation process (502), disparate data sources (1008) includes receiving (1002), from a user, a selection (1004) of the disparate data source. A user is typically a person using a data management a data rendering system to manage and render data of disparate data types (402, 408) from disparate data sources (1008) according to the present invention. Receiving (1002), from a user, a selection (1004) of the disparate data source may be carried out by receiving, through a user interface of a data management and data rendering application, from the user a user instruction containing a selection of the disparate data source and identifying (1009), to the aggregation process (502), the disparate data source (404, 522) in dependence upon the selection (1004). A user instruction is an event received in response to an act by a user such as an event created as a result of a user entering a combination of keystrokes, using a keyboard or keypad, receiving speech from a user, receiving an clicking on icons on a visual display by using a mouse, pressing an icon on a touchpad, or other use act as will occur to those of skill in the art. A user interface in a data management and data rendering application may usefully provide a vehicle for receiving user selections of particular disparate data sources.

In the example of Figure 7, identifying disparate data sources to an aggregation process is carried out by a user. Identifying disparate data sources may also be carried out by processes that require limited or no user interaction. For further explanation, Figure 8 sets forth a flow chart illustrating an exemplary method for aggregating data of disparate data types from disparate data sources requiring little or no user action that includes identifying (1006), to the aggregation process (502), disparate data sources (1008) includes identifying (1102), from a request for data (508), data type information (1106). Disparate data types identify data of different kind and form. That is, disparate data types are data of different kinds. The distinctions in data that define the disparate data types may include a difference in data structure, file format, protocol in which the data is transmitted, and other distinctions as will occur to those of skill in the art. Data type information (1106) is information representing these distinctions in data that define the disparate data types.

Identifying (1102), from the request for data (508), data type information (1106) according to the method of Figure 8 may be carried out by extracting a data type code from the request for data. Alternatively, identifying (1102), from the request for data (508), data type information (1106) may be carried out by inferring the data type of the data being requested from the request itself, such as by extracting data elements from the request and inferring from those data elements the data type of the requested data, or in other ways as will occur to those of skill in the art.

In the method for aggregating of Figure 8, identifying (1006), to the aggregation process (502), disparate data sources also includes identifying (1110), from a data source table (1104), sources of data corresponding to the data type (1116). A data source table is a table containing identification of disparate data sources indexed by the data type of the data retrieved from those disparate data sources. Identifying (1110), from a data source table (1104), sources of data corresponding to the data type (1116) may be carried out by performing a lookup on the data source table in dependence upon the identified data type.

In some cases no such data source may be found for the data type or no such data source table is available for identifying a disparate data

source. In the method of Figure 8 therefore includes an alternative method for identifying (1006), to the aggregation process (502), disparate data sources that includes searching (1108), in dependence upon the data type information (1106), for a data source and identifying (1114), from search results (1112) returned in the data source search, sources of data corresponding to the data type (1116). Searching (1108), in dependence upon the data type information (1106), for a data source may be carried out by creating a search engine query in dependence upon the data type information and querying the search engine with the created query. Querying a search engine may be carried out through the use of URL encoded data passed to a search engine through, for example, an HTTP GET or HTTP POST function. URL encoded data is data packaged in a URL for data communications, in this case, passing a query to a search engine. In the case of HTTP communications, the HTTP GET and POST functions are often used to transmit URL encoded data. In this context, it is useful to remember that URLs do more than merely request file transfers. URLs identify resources on servers. Such resources may be files having filenames, but the resources identified by URLs also include, for example, queries to databases. Results of such queries do not necessarily reside in files, but they are nevertheless data resources identified by URLs and identified by a search engine and query data that produce such resources. An example of URL encoded data is:

`http://www.example.com/search?field1=value1&field2=value2`

This example of URL encoded data representing a query that is submitted over the web to a search engine. More specifically, the example above is a URL bearing encoded data representing a query to a search engine and the query is the string "field1=value1&field2=value2." The exemplary encoding method is to string field names and field values separated by '&' and '=' and designate the encoding as a query by including "search" in the URL. The exemplary URL encoded search query is for explanation and not for limitation. In fact, different search engines may use different syntax in representing a query in a data encoded URL and therefore the particular syntax of the data encoding may vary according to the particular search engine queried.

Identifying (1114), from search results (1112) returned in the data source search, sources of data corresponding to the data type (1116) may

be carried out by retrieving URLs to data sources from hyperlinks in a search results page returned by the search engine.

As discussed above, data management and data rendering for disparate data types includes synthesizing aggregated data of disparate data types into data of a uniform data type. For further explanation, Figure 9 sets forth a flow chart illustrating a method for synthesizing (414) aggregated data of disparate data types (412) into data of a uniform data type. As discussed above, aggregated data of disparate data types (412) is the accumulation, in a single location, of data of disparate types. This location of the aggregated data may be either physical, such as, for example, on a single computer containing aggregated data, or logical, such as, for example, a single interface providing access to the aggregated data. Also as discussed above, disparate data types are data of different kind and form. That is, disparate data types are data of different kinds. Data of a uniform data type is data having been created or translated into a format of predetermined type. That is, uniform data types are data of a single kind that may be rendered on a device capable of rendering data of the uniform data type. Synthesizing (414) aggregated data of disparate data types (412) into data of a uniform data type advantageously makes the content of the disparate data capable of being rendered on a single device.

In the method of Figure 9, synthesizing (414) aggregated data of disparate data types (412) into data of a uniform data type includes receiving (612) aggregated data of disparate data types. Receiving (612) aggregated data of disparate data types (412) may be carried out by receiving, from aggregation process having accumulated the disparate data, data of disparate data types from disparate sources for synthesizing into a uniform data type.

In the method for synthesizing of Figure 9, synthesizing (414) the aggregated data (406) of disparate data types (610) into data of a uniform data type also includes translating (614) each of the aggregated data of disparate data types (610) into text (617) content and markup (619) associated with the text content. Translating (614) each of the aggregated data of disparate data types (610) into text (617) content and markup (619) associated with the text content according to the method of Figure 9 includes representing in text and markup the content of the

aggregated data such that a browser capable of rendering the text and markup may render from the translated data the same content contained in the aggregated data prior to being synthesized.

5 In the method of Figure 9, translating (614) each of the aggregated data of disparate data types (610) into text (617) content and markup (619) may be carried out by creating an X+V document for the aggregated data including text, markup, grammars and so on as will be discussed in more detail below with reference to Figure 10. The use of X+V is for
10 explanation and not for limitation. In fact, other markup languages may be useful in synthesizing (414) the aggregated data (406) of disparate data types (610) into data of a uniform data type according to the present invention such as XML, VXML, or any other markup language as will occur to those of skill in the art.

15 Translating (614) each of the aggregated data of disparate data types (610) into text (617) content and markup (619) such that a browser capable of rendering the text and markup may render from the translated data the same content contained in the aggregated data prior to being
20 synthesized may include augmenting the content in translation in some way. That is, translating aggregated data types into text and markup may result in some modification to the content of the data or may result in deletion of some content that cannot be accurately translated. The quantity of such modification and deletion will vary according to the type of data
25 being translated as well as other factors as will occur to those of skill in the art. Translating (614) each of the aggregated data of disparate data types (610) into text (617) content and markup (619) associated with the text content may be carried out by translating the aggregated data into text and markup and parsing the translated content dependent upon
30 data type. Parsing the translated content dependent upon data type means identifying the structure of the translated content and identifying aspects of the content itself, and creating markup (619) representing the identified structure and content.

35 Consider for further explanation the following markup language depiction of a snippet of audio clip describing the president.

```
<head> original file type= 'MP3' keyword = 'president' number = '50',  
keyword = 'air force' number = '1' keyword = 'white house' number = '2' >  
</head>
```

```
<content>  
    Some content about the president  
</content>
```

5 In the example above an MP3 audio file is translated into text and
markup. The header in the example above identifies the translated data as
having been translated from an MP3 audio file. The exemplary header also
includes keywords included in the content of the translated document and
the frequency with which those keywords appear. The exemplary translated
10 data also includes content identified as 'some content about the
president.'

As discussed above, one useful uniform data type for synthesized
data is XHTML plus Voice. XHTML plus Voice ('X+V') is a Web markup
15 language for developing multimodal applications, by enabling voice with
voice markup. X + V provides voice-based interaction in devices using
both voice and visual elements. Voice enabling the synthesized data for
data management and data rendering according to embodiments of the present
invention is typically carried out by creating grammar sets for the text
20 content of the synthesized data. A grammar is a set of words that may be
spoken, patterns in which those words may be spoken, or other language
elements that define the speech recognized by a speech recognition engine.
Such speech recognition engines are useful in a data management and
rendering engine to provide users with voice navigation of and voice
25 interaction with synthesized data.

Further explanation, therefore, Figure 10 sets forth a flow chart
illustrating a method for synthesizing (414) aggregated data of disparate
data types (412) into data of a uniform data type that includes
30 dynamically creating grammar sets for the text content of synthesized data
for voice interaction with a user. Synthesizing (414) aggregated data of
disparate data types (412) into data of a uniform data type according to
the method of Figure 10 includes receiving (612) aggregated data of
disparate data types (412). As discussed above, receiving (612)
35 aggregated data of disparate data types (412) may be carried out by
receiving, from aggregation process having accumulated the disparate data,
data of disparate data types from disparate sources for synthesizing into
a uniform data type.

The method of Figure 10 for synthesizing (414) aggregated data of disparate data types (412) into data of a uniform data type also includes translating (614) each of the aggregated data of disparate data types (412) into translated data (1204) comprising text content and markup associated with the text content. As discussed above, translating (614) each of the aggregated data of disparate data types (412) into text content and markup associated with the text content includes representing in text and markup the content of the aggregated data such that a browser capable of rendering the text and markup may render from the translated data the same content contained in the aggregated data prior to being synthesized. In some cases, translating (614) the aggregated data of disparate data types (412) into text content and markup such that a browser capable of rendering the text and markup may include augmenting or deleting some of the content being translated in some way as will occur to those of skill in the art.

In the method of Figure 10, translating (1202) each of the aggregated data of disparate data types (412) into translated data (1204) comprising text content and markup may be carried out by creating an X+V document for the synthesized data including text, markup, grammars and so on as will be discussed in more detail below. The use of X+V is for explanation and not for limitation. In fact, other markup languages may be useful in translating (614) each of the aggregated data of disparate data types (412) into translated data (1204) comprising text content and markup associated with the text content as will occur to those of skill in the art.

The method of Figure 10 for synthesizing (414) aggregated data of disparate data types (412) into data of a uniform data type may include dynamically creating (1206) grammar sets (1216) for the text content. As discussed above, a grammar is a set of words that may be spoken, patterns in which those words may be spoken, or other language elements that define the speech recognized by a speech recognition engine.

In the method of Figure 10, dynamically creating (1206) grammar sets (1216) for the text content also includes identifying (1208) keywords (1210) in the translated data (1204) determinative of content or logical structure and including the identified keywords in a grammar associated with the translated data. Keywords determinative of content are words and

phrases defining the topics of the content of the data and the information presented the content of the data. Keywords determinative of logical structure are keywords that suggest the form in which information of the content of the data is presented. Examples of logical structure include typographic structure, hierarchical structure, relational structure, and other logical structures as will occur to those of skill in the art.

Identifying (1208) keywords (1210) in the translated data (1204) determinative of content may be carried out by searching the translated text for words that occur in a text more often than some predefined threshold. The frequency of the word exceeding the threshold indicates that the word is related to the content of the translated text because the predetermined threshold is established as a frequency of use not expected to occur by chance alone. Alternatively, a threshold may also be established as a function rather than a static value. In such cases, the threshold value for frequency of a word in the translated text may be established dynamically by use of a statistical test which compares the word frequencies in the translated text with expected frequencies derived statistically from a much larger corpus. Such a larger corpus acts as a reference for general language use.

Identifying (1208) keywords (1210) in the translated data (1204) determinative of logical structure may be carried out by searching the translated data for predefined words determinative of structure. Examples of such words determinative of logical structure include 'introduction,' 'table of contents,' 'chapter,' 'stanza,' 'index,' and many others as will occur to those of skill in the art.

In the method of Figure 10, dynamically creating (1206) grammar sets (1216) for the text content also includes creating (1214) grammars in dependence upon the identified keywords (1210) and grammar creation rules (1212). Grammar creation rules are a pre-defined set of instructions and grammar form for the production of grammars. Creating (1214) grammars in dependence upon the identified keywords (1210) and grammar creation rules (1212) may be carried out by use of scripting frameworks such as JavaServer Pages, Active Server Pages, PHP, Perl, XML from translated data. Such dynamically created grammars may be stored externally and referenced, in for example, X+V the <grammar src=""/> tag that is used to reference external grammars.

The method of Figure 10 for synthesizing (414) aggregated data of disparate data types (412) into data of a uniform data type includes associating (1220) the grammar sets (1216) with the text content. Associating (1220) the grammar sets (1216) with the text content includes inserting (1218) markup (1224) defining the created grammar into the translated data (1204). Inserting (1218) markup in the translated data (1204) may be carried out by creating markup defining the dynamically created grammar inserting the created markup into the translated document.

The method of Figure 10 also includes associating (1222) an action (420) with the grammar. As discussed above, an action is a set of computer instructions that when executed carry out a predefined task. Associating (1222) an action (420) with the grammar thereby provides voice initiation of the action such that the associated action is invoked in response to the recognition of one or more words or phrases of the grammar.

As discussed above, data management and data rendering for disparate data types includes identifying an action in dependence upon the synthesized data. For further explanation, Figure 11 sets forth a flow chart illustrating an exemplary method for identifying an action in dependence upon the synthesized data (416) including receiving (616) a user instruction (620) and identifying an action in dependence upon the synthesized data (416) and the user instruction. In the method of Figure 11, identifying an action may be carried out by retrieving an action ID from an action list. In the method of Figure 11, retrieving an action ID from an action list includes retrieving from a list the identification of the action (the 'action ID') to be executed in dependence upon the user instruction and the synthesized data. The action list can be implemented, for example, as a Java list container, as a table in random access memory, as a SQL database table with storage on a hard drive or CD ROM, and in other ways as will occur to those of skill in the art. As mentioned above, the actions themselves comprise software, and so can be implemented as concrete action classes embodied, for example, in a Java package imported into a data management and data rendering module at compile time and therefore always available during run time.

In the method of Figure 11, receiving (616) a user instruction (620) includes receiving (1504) speech (1502) from a user, converting (1506) the

speech (1502) to text (1508); determining (1512) in dependence upon the text (1508) and a grammar (1510) the user instruction (620) and determining (1602) in dependence upon the text (1508) and a grammar (1510) a parameter (1604) for the user instruction (620). As discussed above with reference to Figure 4, a user instruction is an event received in response to an act by a user. A parameter to a user instruction is additional data further defining the instruction. For example, a user instruction for 'delete email' may include the parameter 'August 11, 2005' defining that the email of August 11, 2005 is the synthesized data upon which the action invoked by the user instruction is to be performed. Receiving (1504) speech (1502) from a user, converting (1506) the speech (1502) to text (1508); determining (1512) in dependence upon the text (1508) and a grammar (1510) the user instruction (620); and determining (1602) in dependence upon the text (1508) and a grammar (1510) a parameter (1604) for the user instruction (620) may be carried out by a speech recognition engine incorporated into a data management and data rendering module according to the present invention.

Identifying an action in dependence upon the synthesized data (416) according to the method of Figure 11 also includes selecting (618) synthesized data (416) in response to the user instruction (620). Selecting (618) synthesized data (416) in response to the user instruction (620) may be carried out by selecting synthesized data identified by the user instruction (620). Selecting (618) synthesized data (416) may also be carried out by selecting the synthesized data (416) in dependence upon a parameter (1604) of the user instruction (620).

Selecting (618) synthesized data (416) in response to the user instruction (620) may be carried out by selecting synthesized data context information (1802). Context information is data describing the context in which the user instruction is received such as, for example, state information of currently displayed synthesized data, time of day, day of week, system configuration, properties of the synthesized data, or other context information as will occur to those of skill in the art. Context information may be usefully used instead or in conjunction with parameters to the user instruction identified in the speech. For example, the context information identifying that synthesized data translated from an email document is currently being displayed may be used to supplement the

speech user instruction 'delete email' to identify upon which synthesized data to perform the action for deleting an email.

5 Identifying an action in dependence upon the synthesized data (416) according to the method of Figure 11 also includes selecting (624) an action (420) in dependence upon the user instruction (620) and the selected data (622). Selecting (624) an action (420) in dependence upon the user instruction (620) and the selected data (622) may be carried out by selecting an action identified by the user instruction. Selecting
10 (624) an action (420) may also be carried out by selecting the action (420) in dependence upon a parameter (1604) of the user instructions (620) and by selecting the action (420) in dependence upon a context information (1802). In the example of Figure 11, selecting (624) an action (420) is carried out by retrieving an action from an action database (1105) in
15 dependence upon one or more a user instructions, parameters, or context information.

Executing the identified action may be carried out by use of a switch() statement in an action agent of a data management and data
20 rendering module. Such a switch() statement can be operated in dependence upon the action ID and implemented, for example, as illustrated by the following segment of pseudocode:

```
switch (actionID) {  
25   Case 1:  actionNumber1.take_action();  break;  
   Case 2:  actionNumber2.take_action();  break;  
   Case 3:  actionNumber3.take_action();  break;  
   Case 4:  actionNumber4.take_action();  break;  
   Case 5:  actionNumber5.take_action();  break;  
30   // and so on  
} // end switch()
```

The exemplary switch statement selects an action to be performed on synthesized data for execution depending on the action ID. The tasks
35 administered by the switch() in this example are concrete action classes named actionNumber1, actionNumber2, and so on, each having an executable member method named 'take_action(),' which carries out the actual work implemented by each action class.

Executing an action may also be carried out in such embodiments by use of a hash table in an action agent of a data management and data rendering module. Such a hash table can store references to action object keyed by action ID, as shown in the following pseudocode example. This
5 example begins by an action service's creating a hashtable of actions, references to objects of concrete action classes associated with a user instruction. In many embodiments it is an action service that creates such a hashtable, fills it with references to action objects pertinent to a particular user instruction, and returns a reference to the hashtable to
10 a calling action agent.

```
Hashtable ActionHashTable = new Hashtable();  
ActionHashTable.put("1", new Action1());  
ActionHashTable.put("2", new Action2());  
15 ActionHashTable.put("3", new Action3());
```

Executing a particular action then can be carried out according to the following pseudocode:

```
20 Action anAction = (Action) ActionHashTable.get("2");  
if (anAction != null) anAction.take_action();
```

Executing an action may also be carried out by use of list. Lists often function similarly to hashtables. Executing a particular action,
25 for example, can be carried out according to the following pseudocode:

```
List ActionList = new List();  
ActionList.add(1, new Action1());  
ActionList.add(2, new Action2());  
30 ActionList.add(3, new Action3());
```

Executing a particular action then can be carried out according to the following pseudocode:

```
35 Action anAction = (Action) ActionList.get(2);  
if (anAction != null) anAction.take_action();
```

The three examples above use switch statements, hash tables, and list objects to explain executing actions according to embodiments of the

present invention. The use of switch statements, hash tables, and list objects in these examples are for explanation, not for limitation. In fact, there are many ways of executing actions according to embodiments of the present invention, as will occur to those of skill in the art, and all such ways are well within the scope of the present invention.

For further explanation of identifying an action in dependence upon the synthesized data consider the following example of user instruction that identifies an action, a parameter for the action, and the synthesized data upon which to perform the action. A user is currently viewing synthesized data translated from email and issues the following speech instruction: "Delete email dated August 15, 2005." In the current example, identifying an action in dependence upon the synthesized data is carried out by selecting an action to delete and synthesized data in dependence upon the user instruction, by identifying a parameter for the delete email action identifying that only one email is to be deleted, and by selecting synthesized data translated from the email of August 15, 2005 in response to the user instruction.

For further explanation of identifying an action in dependence upon the synthesized data consider the following example of user instruction that does not specifically identify the synthesized data upon which to perform an action. A user is currently viewing synthesized data translated from a series of emails and issues the following speech instruction: "Delete current email." In the current example, identifying an action in dependence upon the synthesized data is carried out by selecting an action to delete synthesized data in dependence upon the user instruction. Selecting synthesized data upon which to perform the action, however, in this example is carried out in dependence upon the following data selection rule that makes use of context information.

```
If synthesized data = displayed;  
    Then synthesized data = 'current'.
```

```
If synthesized includes = email type code;  
    Then synthesized data = email.
```

The exemplary data selection rule above identifies that if synthesized data is displayed then the displayed synthesized data is

'current' and if the synthesized data includes an email type code then the synthesized data is email. Context information is used to identify currently displayed synthesized data translated from an email and bearing an email type code. Applying the data selection rule to the exemplary
5 user instruction "delete current email" therefore results in deleting currently displayed synthesized data having an email type code.

As discussed above, data management and data rendering for disparate data types often includes channelizing the synthesized data. Channelizing the synthesized data (416) advantageously results in the separation of
10 synthesized data into logical channels. A channel implemented as a logical accumulation of synthesized data sharing common attributes having similar characteristics. Examples of such channels are 'entertainment channel' for synthesized data relating to entertainment, 'work channel' for synthesized data relating to work, 'family channel' for synthesized
15 data relating to a user's family and so on.

For further explanation, therefore, Figure 12 sets forth a flow chart illustrating an exemplary method for channelizing (422) the synthesized data (416) according to embodiments of the present invention,
20 which includes identifying (802) attributes of the synthesized data (804). Attributes of synthesized data (804) are aspects of the data which may be used to characterize the synthesized data (416). Exemplary attributes (804) include the type of the data, metadata present in the data, logical structure of the data, presence of particular keywords in the content of
25 the data, the source of the data, the application that created the data, URL of the source, author, subject, date created, and so on. Identifying (802) attributes of the synthesized data (804) may be carried out by comparing contents of the synthesized data (804) with a list of predefined attributes. Another way that identifying (802) attributes of the
30 synthesized data (804) may be carried out by comparing metadata associated with the synthesized data (804) with a list of predefined attributes.

The method of Figure 12 for channelizing (422) the synthesized data (416) also includes characterizing (808) the attributes of the synthesized
35 data (804). Characterizing (808) the attributes of the synthesized data (804) may be carried out by evaluating the identified attributes of the synthesized data. Evaluating the identified attributes of the synthesized data may include applying a characterization rule (806) to an identified

attribute. For further explanation consider the following
characterization rule:

If synthesized data = email; AND

If email to = "Joe"; AND

If email from = "Bob";

Then email = 'work email.'

In the example above, the characterization rule dictates that if
synthesized data is an email and if the email was sent to "Joe" and if the
email sent from "Bob" then the exemplary email is characterized as a 'work
email.'

Characterizing (808) the attributes of the synthesized data (804)
may further be carried out by creating, for each attribute identified, a
characteristic tag representing a characterization for the identified
attribute. Consider for further explanation the following example of
synthesized data translated from an email having inserted within it a
characteristic tag.

<head >

original message type = 'email' to = 'joe' from = 'bob' re = 'I will
be late tomorrow'</head>

<characteristic>

characteristic = 'work'

<characteristic>

<body>

Some body content

</body>

In the example above, the synthesized data is translated from an
email sent to Joe from 'Bob' having a subject line including the text 'I
will be late tomorrow. In the example above <characteristic> tags
identify a characteristic field having the value 'work' characterizing the
email as work related. Characteristic tags aid in channelizing synthesized

data by identifying characteristics of the data useful in channelizing the data.

The method of Figure 12 for channelizing (422) the synthesized data (416) also includes assigning (814) the data to a predetermined channel (816) in dependence upon the characterized attributes (810) and channel assignment rules (812). Channel assignment rules (812) are predetermined instructions for assigning synthesized data (416) into a channel in dependence upon characterized attributes (810). Consider for further explanation the following channel assignment rule:

If synthesized data = 'email'; and

If Characterization = 'work related email'

Then channel = 'work channel.'

In the example above, if the synthesized data is translated from an email and if the email has been characterized as 'work related email' then the synthesized data is assigned to a 'work channel.'

Assigning (814) the data to a predetermined channel (816) may also be carried out in dependence upon user preferences, and other factors as will occur to those of skill in the art. User preferences are a collection of user choices as to configuration, often kept in a data structure isolated from business logic. User preferences provide additional granularity for channelizing synthesized data according to the present invention.

Under some channel assignment rules (812), synthesized data (416) may be assigned to more than one channel (816). That is, the same synthesized data may in fact be applicable to more than one channel. Assigning (814) the data to a predetermined channel (816) may therefore be carried out more than once for a single portion of synthesized data.

The method of Figure 12 for channelizing (422) the synthesized data (416) may also include presenting (426) the synthesized data (416) to a user through one or more channels (816). One way presenting (426) the synthesized data (416) to a user through one or more channels (816) may be

carried out is by presenting summaries or headings of available channels in a user interface allowing a user access to the content of those channels. These channels could be accessed via this presentation in order to access the synthesized data (416). The synthesized data is
5 additionally to the user through the selected channels by displaying or playing the synthesized data (416) contained in the channel.

As discussed above, data management and data rendering for disparate data types often includes identifying a data source from which to
10 aggregate data. Accessing data from the identified data source for aggregation may be improved by dynamic creation of user interfaces for data sources that often include display text describing information about the data source and grammars speech enabling the user interfaces.

For further explanation, therefore, Figure 13 sets forth a flow chart illustrating an exemplary method for dynamic creation of user interfaces for data management and data rendering according to embodiments of the present invention. The phrase 'user interface' is used to mean computer software components for communications between a user and a
20 computer software application, so as to allow input from the user to the computer software application and output from the computer software application to the user. Often user interfaces are implemented as one or more user interface objects, which are software components designed for receiving user instructions invoking an action, such as retrieving data
25 from a data source. A user interface object is often used to access data from the data source. A user interface object may also be used to represent a data source for aspects of data management and data rendering such as managing aggregation and synthesis preferences and priorities. Such user interface objects may be implemented as GUI objects invoked by
30 direct manipulation from a user such as text boxes, windows, menus, radio buttons, check boxes, and icons. User interface objects may also be voice recognition objects having associated grammars invoked by user speech, and other user interface objects as will occur to those of skill in the art.

The method of Figure 13 includes receiving (302) an identification
35 of a data source (304). As mentioned above, a data source may be one of many disparate data sources. Such data sources may be any device or network location capable of providing access to data. Examples of data sources include servers serving up files, web sites, cellular phones,

PDAs, MP3 players, and so on as will occur to those of skill in the art. An identification of a data source (304) may include, for example, a URI identifying the data source, a file name, a file path, a pointer to a location in memory, or any other identification of a data source as will occur to those of skill in the art.

Receiving (302) an identification of a data source (304) according to the method of Figure 13 may be carried out by receiving, from a user, an identification of a data source through, for example, a user interface designed to receive such data source identification from users. A user interface designed to receive such data source identification from users is often capable of accepting a user identification through text boxes in the user interface. A user may populate the text box by copying a URL from an address window in a browser pointed to the data source and pasting the URL into the text box. Some browsers and user interfaces support drag-and-drop features facilitating the population of text boxes. Such drag-and-drop features allow a user to populate a text box by dragging a hyperlink from a page displayed by the browser or dragging a URL from an address window in a browser pointed to the data source and dropping the hyperlink or URL in the text box.

Many GUI software toolkits support developing user interfaces capable receiving (302) an identification of a data source (304) from a user through drag and drop features. Examples of such GUI software toolkits include, for example, Java Foundation Classes ('JFC') toolkits, including the Abstract Windowing Toolkit (AWT), Swing, and Java2D. Together, the JFC toolkits provide a consistent user interface for Java programs, regardless of the underlying user interface system. Swing, for example, supports dragging and dropping an identification of a data source into a text box in a user interface by instantiating a member of the TransferHandler object class for both the identification of the data source and the text box. The TransferHandler for the identification of the data source manages the transfer of the identification of the data source to the text box by creating Transferable object containing the identification of the data source. The TransferHandler for the text box then extracts the identification of the data source from the Transferable object.

The method of Figure 13 for dynamic creation of user interfaces for data management and data rendering also includes retrieving (308) data from the identified data source (314). Often the data retrieved from the identified data source is a web page (306) hosted on a server (316) as illustrated in Figure 13. In the method of Figure 13, retrieving (308) data from the identified data source (314) includes sending (312) an HTTP request (313) for the web page (306) to a server (316) and receiving (310) an HTTP response (315) containing the web page (306) contents from the server (316).

In the example of Figure 13, the data retrieved from the identified data source is a web page (306) hosted on a server (316). This is for explanation and not for limitation. In fact, the data retrieved from the identified data source may be of many different data types and all such data types are within the scope of the present invention.

The method of Figure 13 also includes dynamically creating (318), in dependence upon the retrieved data (309), a user interface object (324) to access the identified data source (314). As discussed above, user interface objects are software components designed for receiving user instructions invoking an action, such as retrieving data from a data source. Such user interface objects may be implemented as GUI objects invoked by direct manipulation from a user such as text boxes, windows, menus, radio buttons, check boxes, and icons. User interface objects may also be voice recognition objects having associated grammars invoked by user speech, and other user interface objects as will occur to those of skill in the art.

In the method of Figure 13, dynamically creating (318), in dependence upon the retrieved data (309), a user interface object (324) to access the identified data source (314) include creating (320) display text (334) describing source information. Source information is information describing the data source, content available from the data source, and other information about the data source as will occur to those of skill in the art. Examples of source information include the title of a web page, the domain name of a data source, the author of a file available at the data source, keywords or summaries derived from the content of the data available from the data source, and other source information as will occur to those of skill in the art.

Display text (334) describing source information is text which may be presented to the user visually through, for example, a browser, as speech rendered by a speech engine, or in other ways as will occur to those of skill in the art. Creating display text (334) describing source information may be carried out by analyzing the retrieved data (309) to extract the source information and creating display text (334) describing the extracted source information. Analyzing the retrieved data (309) to extract the source information may be carried out by identifying keywords in the retrieved data, extracting metadata from the retrieved data such as header information in a web page, identifying titles in the retrieved data, identifying abstracts or summaries in the retrieved data and in other ways as will occur to those of skill in the art .

For further illustration, consider the example of a web page written in XML retrieved from an identified data source, (314) according to the method for dynamic creation of user interfaces for data management and data rendering according to the method of Figure 13. The retrieved XML web page contains the following text and markup:

```
<head >
  <title>somemenu.com</title>
  <description>somemenu.com has information about your
community.</description>
</head>.
```

The XML page includes text and markup containing information about the web site and about the contents of the data available from that site. The exemplary XML page represented above, for example, contains a header, which gives information about the type of data provided by the data source. The header in the exemplary XML page contains "<title>" and "</title>" tags identifying the title of the data source as somemenu.com. The header in the exemplary XML page contains "<description>" and "</description>" tags identifying the text "somemenu.com has information about your community," which gives a description of the content of the identified data source (314). Creating (320) display text (334) for a user interface object (324) to access the exemplary XML web page according to the method of Figure 13 may be carried out by using the information included in the XML header information to create display text, such as by

creating display text reciting the description "somemenu.com has information about your community."

In the method of Figure 13, dynamically creating (318), in dependence upon the retrieved data (309), a user interface object (324) to access the identified data source (314) also includes creating (322) grammars (336) to voice enable the user interface object (324). As discussed above, a grammar (336) is a set of words that may be spoken, patterns in which those words may be spoken, or other language elements that define the speech recognized by a speech recognition engine. Grammars (336) which voice enable the user interface object (324) may be invoked by speech to select the identified data source (314) represented by the user interface object (324) and retrieve data from the identified data source (314).

Creating (322) grammars (336) to voice enable the user interface object (324) may be carried out by analyzing the retrieved data (309) to extract words and phrases for inclusion in the grammar. The extracted words and phrases for inclusion in the grammar are often identified through the use of grammar creation rules. Grammar creation rules are a pre-defined set of instructions and grammar form for the production of grammars. Creating such grammars may be carried out by use of scripting frameworks such as JavaServer Pages, Active Server Pages, PHP, Perl.

For further illustration, consider the same example of a web page written in XML retrieved from an identified data source (314) according to the method for dynamic creation of user interfaces for data management and data rendering according to the method of Figure 13. The retrieved XML web page contains the following text and markup:

```
<head >  
  <title>somemenu.com</title>  
  <description>somemenu.com has information about your  
  community.</description>  
</head>.
```

The XML page includes text and markup containing information about the web site and about the contents of the data available from that site. The exemplary XML page represented above, for example, contains a header,

which gives information about the type of data provided by the data source. The header in the exemplary XML page contains "<title>" and "</title>" tags identifying the title of the data source as somemenu.com. The header in the exemplary XML page contains "<description>" and
5 "</description>" tags identifying the text "somemenu.com has a information about your community," which gives a description of the content of the identified data source (314). Creating (322) grammars (336) to voice enable the user interface object (324) to access the exemplary XML web page according to the method of Figure 13 may be carried out by extracting
10 words included in the XML header information, such as the keywords "community" and "somemenu.com" and including the extracted words in the grammar.

The method of Figure 13 also includes assigning (328) the identified
15 data source (314) to a channel (332). Assigning (328) the identified data source (314) to a channel (332) may include identifying attributes of the retrieved data (309) from the identified data source (314). Attributes of the retrieved data (309) from the identified data source (314) are aspects of the retrieved data (309) which may be used to characterize the
20 identified data source (314). Exemplary attributes include the type of the data available at the data source, metadata present in the data, logical structure of the data, presence of particular keywords in the content of the data, the application that created the data, author, subject, date created, and so on. Identifying attributes of the retrieved
25 data (309) may be carried out by comparing the retrieved data (309) with a list of predefined attributes.

In the method of Figure 13, assigning (328) the identified data source (314) to a channel (332) may also include characterizing the
30 identified attributes of retrieved data (309). Characterizing the identified attributes of retrieved data may be carried out applying a characterization rule to an identified attribute. For further explanation consider the following characterization rule:

35 If title contains the word 'patent'; AND

If content contains the words 'federal circuit'

Then source = 'work source' AND

Channel = 'work channel.'

5 In the example above, the characterization rule dictates that if the retrieved data (309) from the identified data source (314) contains the word "patent" in markup denoting a title and the words "federal circuit" in the content of the source, then the exemplary identified data source (314) is characterized as a 'work source,' a data source useful to the user at work. The data source is assigned to the 'work channel.'

10 Under some channel assignment rules, an identified data source (314) may be assigned to more than one channel (332). That is, the same identified data source (314) may, in fact, be assigned to more than one channel. Assigning (326) the identified data source (314) to channel (332) may therefore be carried out more than once for a single identified
15 data source (314).

The method of Figure 13 for assigning (328) the identified data source (314) to a channel (332) may also include presenting the identified data source (314) to a user through one or more channels (332). One way
20 presenting the identified data source (314) to a user through one or more channels (332) may be carried out is by presenting a user interface object (324) associated with the identified data source (314) in a user interface together with other user interface objects representing other identified data sources in the same channel.

25 The method of Figure 13 for assigning (328) the identified data source (314) to a channel (332) may also include assigning (326) the identified data source (314) to a channel (332) in dependence upon other synthesized content in the channel (330). Assigning (326) the identified
30 data source (314) to a channel (332) in dependence upon other synthesized content in the channel (330) may be carried out by comparing synthesized content from other sources already assigned to the channel with content from the identified data source (314) proposed to be assigned to a channel and determining whether the content from the identified data source (314)
35 is sufficiently similar to the synthesized content from other sources already assigned to the channel. If the content from the identified data source (314) is sufficiently similar to the synthesized content from other sources already assigned to the channel then the identified data source is assigned to the channel. If the content from the identified data source

(314) is not sufficiently similar to the synthesized content from other sources already assigned to the channel then the identified data source is not assigned to the channel.

5 For further explanation, Figure 14 sets forth a line drawing of an exemplary web page displaying available data sources and a configuration page designed to receive identifications of data sources and present display text describing source information of the identified data source in accordance with the present invention. In the example of Figure 14, a
10 web browser is displaying a web page (338) with the title "Football Central Website" (339) containing content generally about football. The web page (338) also contains headings concerning specific subjects relating to football, such as the heading titled "Great Games" (344) and the heading titled "How to Throw a Football" (346). The web page (338) is
15 retrieved from the URL (340), "www.someurl.com," which is an identification of the data source serving up the web page.

 The exemplary configuration page (342) of Figure 14 is a user interface capable generally of receiving identifications of data sources
20 in accordance with the present invention. The exemplary configuration page (342) has text box input widgets (356, 348) for receiving from a user an identified data source. A text box input widget is an input widget which accepts an identification of the data source in the form of text such as a URL. In the exemplary configuration page (342) of Figure 14,
25 the user may input the identified data source by drag-and-drop or cut-and-paste from another graphical user interface ('GUI'), such as, for example, the web page (338) displayed in a browser of Figure 14. The user may alternatively input the identified data source by typing the
30 identification of the data source into the text box input widget.

 After inputting an identified data source, the user may submit the identified data source by clicking on the 'submit' button (350) using a mouse. After the identified data source is submitted, dynamically creating user interfaces for data management and data rendering according
35 to the present invention creates a user interface object for the identified data source that includes display text and a grammar. In the example of Figure 14, this display text is visually displayed in a text box widget (354, 353). This display text can be changed by typing in new

display text, thereby changing the text associated with the user interface object.

5 In the example of Figure 14, a user identified the source of the web page (338), "www.someurl.com," as a data source by cutting the text containing the web page's URL (340), "www.someurl.com," from the browser's address bar (341), and pasting the text in a text box input widget (356) in the configuration page (342). In response to receiving the identification of a data source, the web page (338) is retrieved from the
10 identified data source "www.someurl.com," and a user interface object to access the identified data source is created by creating display text describing source information and a grammar to voice enable the user interface object. In the example of Figure 14, display text describing source information recites "Football Central - Great Games" and is
15 displayed in the text box widget (354) below a text box input widget (356) containing the identification of the data source, which, in the example of Figure 14, is a URL (340), "www.someurl.com."

20 Exemplary embodiments of the present invention are described largely in the context of a fully functional computer system for dynamic creation of user interfaces for data management and data rendering. Readers of skill in the art will recognize, however, that the present invention also may be embodied in a computer program product disposed on signal bearing media for use with any suitable data processing system. Such signal
25 bearing media may be transmission media or recordable media for machine-readable information, including magnetic media, optical media, or other suitable media. Examples of recordable media include magnetic disks in hard drives or diskettes, compact disks for optical drives, magnetic tape, and others as will occur to those of skill in the art. Examples of
30 transmission media include telephone networks for voice communications and digital data communications networks such as, for example, Ethernets and networks that communicate with the Internet Protocol and the World Wide Web. Persons skilled in the art will immediately recognize that any computer system having suitable programming means will be capable of
35 executing the steps of the method of the invention as embodied in a program product. Persons skilled in the art will recognize immediately that, although some of the exemplary embodiments described in this specification are oriented to software installed and executing on computer

hardware, nevertheless, alternative embodiments implemented as firmware or as hardware are well within the scope of the present invention.

5 It will be understood from the foregoing description that modifications and changes may be made in various embodiments of the present invention without departing from its scope. The descriptions in this specification are for purposes of illustration only and are not to be construed in a limiting sense. The scope of the present invention is limited only by the language of the following claims.

CLAIMS

1. A computer-implemented method for creation of user interfaces for data management and data rendering, the method comprising:

receiving an identification of a data source;
retrieving data from the identified data source; and
creating, in accordance with the retrieved data, a user interface object operable to access the identified data source.

2. The method of claim 1 wherein the identified data source is a web page; and

retrieving data from the identified data source further comprises:
sending an HTTP request for the web page to a server; and
receiving an HTTP response containing the web page contents from the server.

3. The method of claim 1 wherein:
the creating step further comprises the step of: creating display text associated with describing source information.

4. The method of claim 1 wherein:
the creating step further comprises the step of: creating grammars operable to voice enable the user interface object.

5. The method of claim 1 further comprising the step of:
assigning the identified data source to a channel.

6. The method of claim 5 wherein the identified data source is assigned to a channel in accordance with other synthesized content in the channel.

7. A system for creation of user interfaces for data management and data rendering, the system comprising:

a computer processor;
a computer memory operatively coupled to the computer processor, the computer memory having disposed within it computer program instructions capable of:

receiving an identification of a data source;
retrieving data from the identified data source; and

creating, in accordance the retrieved data, a user interface object operable to access the identified data source.

8. The system of claim 7 wherein the identified data source is a web page; and wherein the computer memory also has disposed within it computer program instructions capable of:

sending an HTTP request for the web page to a server; and

receiving an HTTP response containing the web page contents from the server.

9. The system of claim 7 wherein the computer memory also has disposed within it computer program instructions capable of:

creating, in accordance with the retrieved data, a user interface object to access the identified data source further comprises creating display text associated with describing source information.

10. The system of claim 7 wherein the computer memory also has disposed within it computer program instructions capable of:

creating, in accordance with the retrieved data, a user interface object to access the identified data source further comprises creating grammars operable to voice enable the user interface object.

11. The system of claim 7 wherein the computer memory also has disposed within it computer program instructions capable of assigning the identified data source to a channel.

12. The system of claim 11 wherein the computer memory also has disposed within it computer program instructions capable of assigning the identified data source to a channel in accordance with other synthesized content in the channel.

13. A computer program comprising program code means adapted to perform all the steps of any of claims 1 to 6 when said program is run on a computer.

1/14

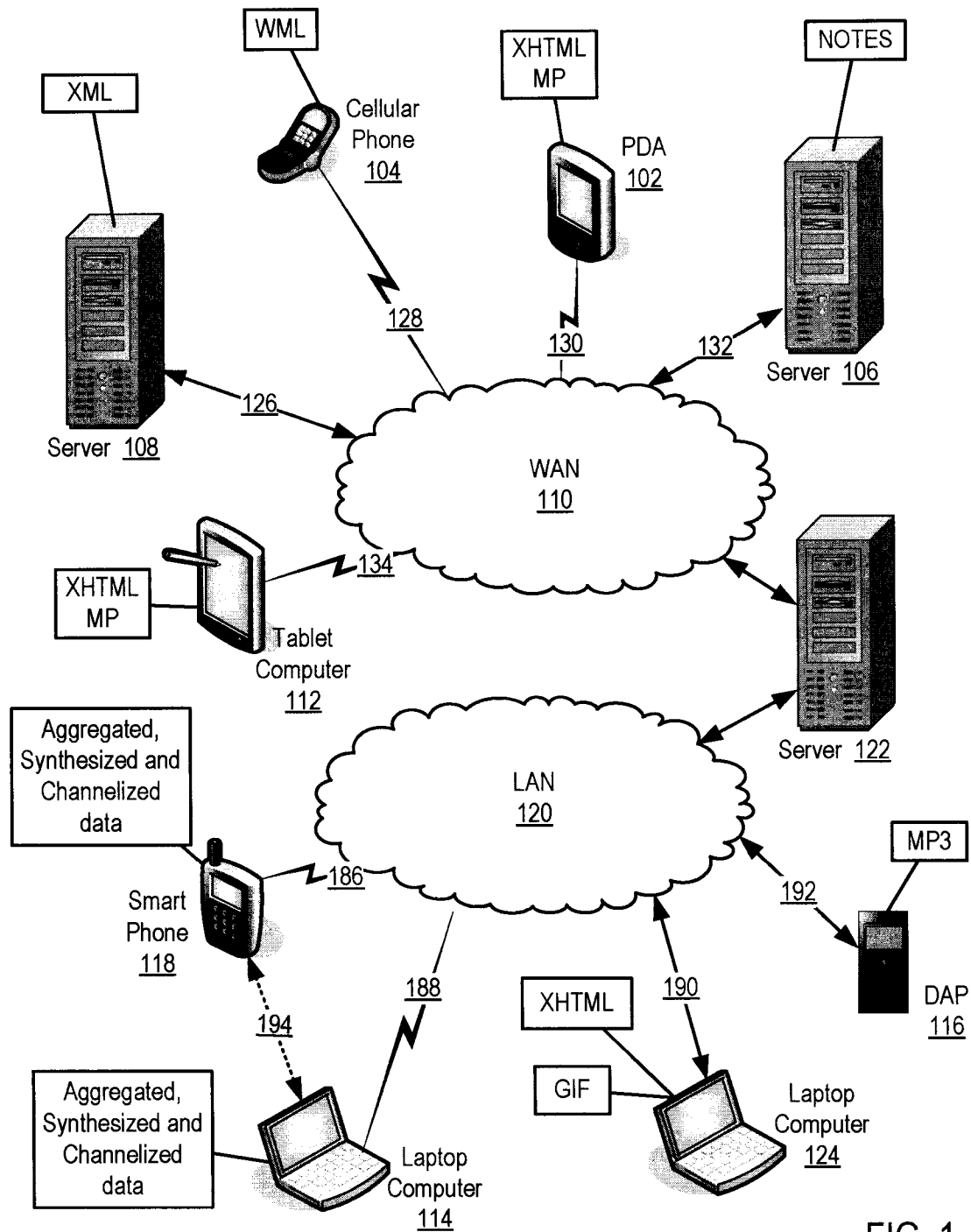


FIG. 1

2/14

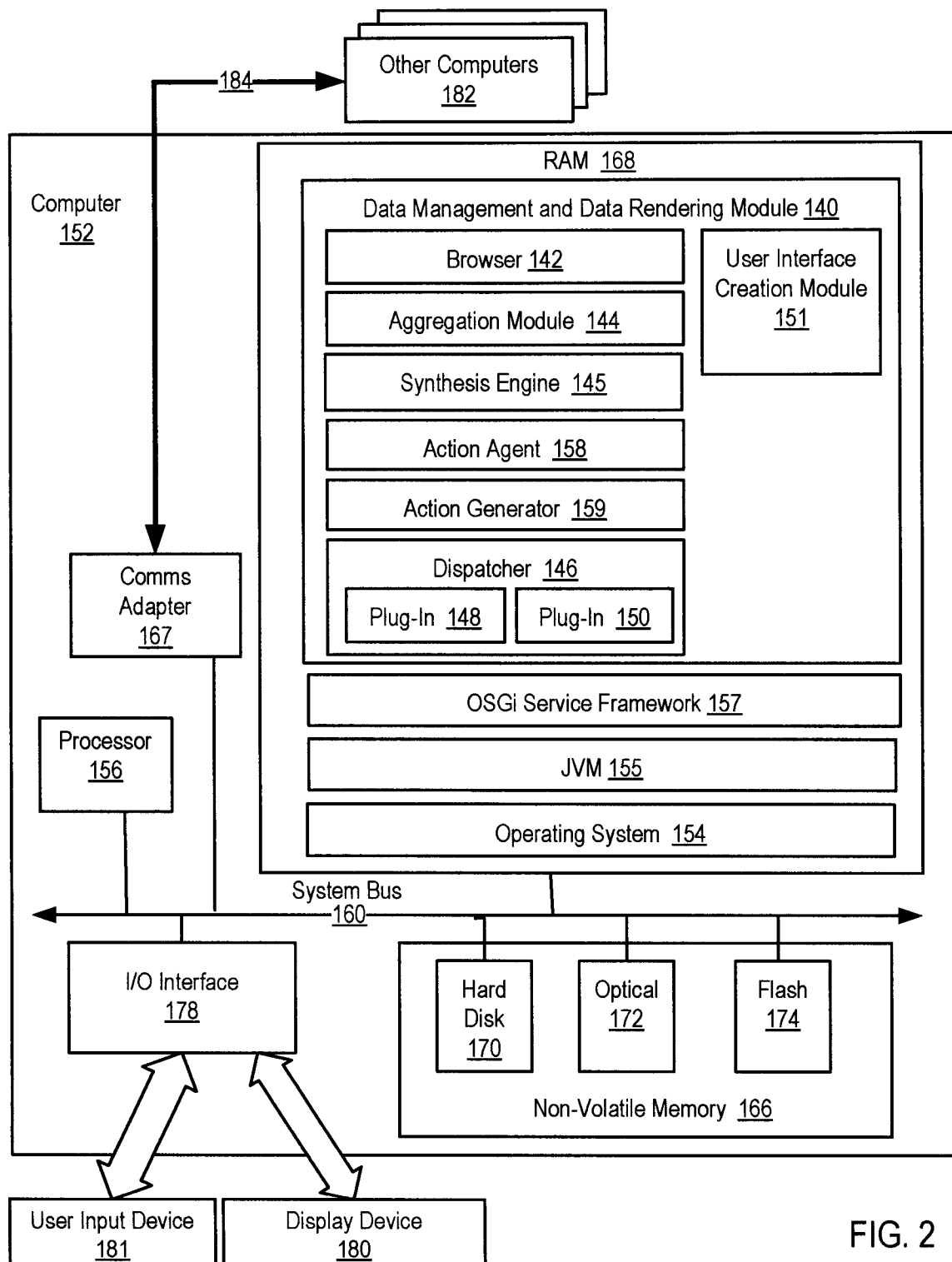
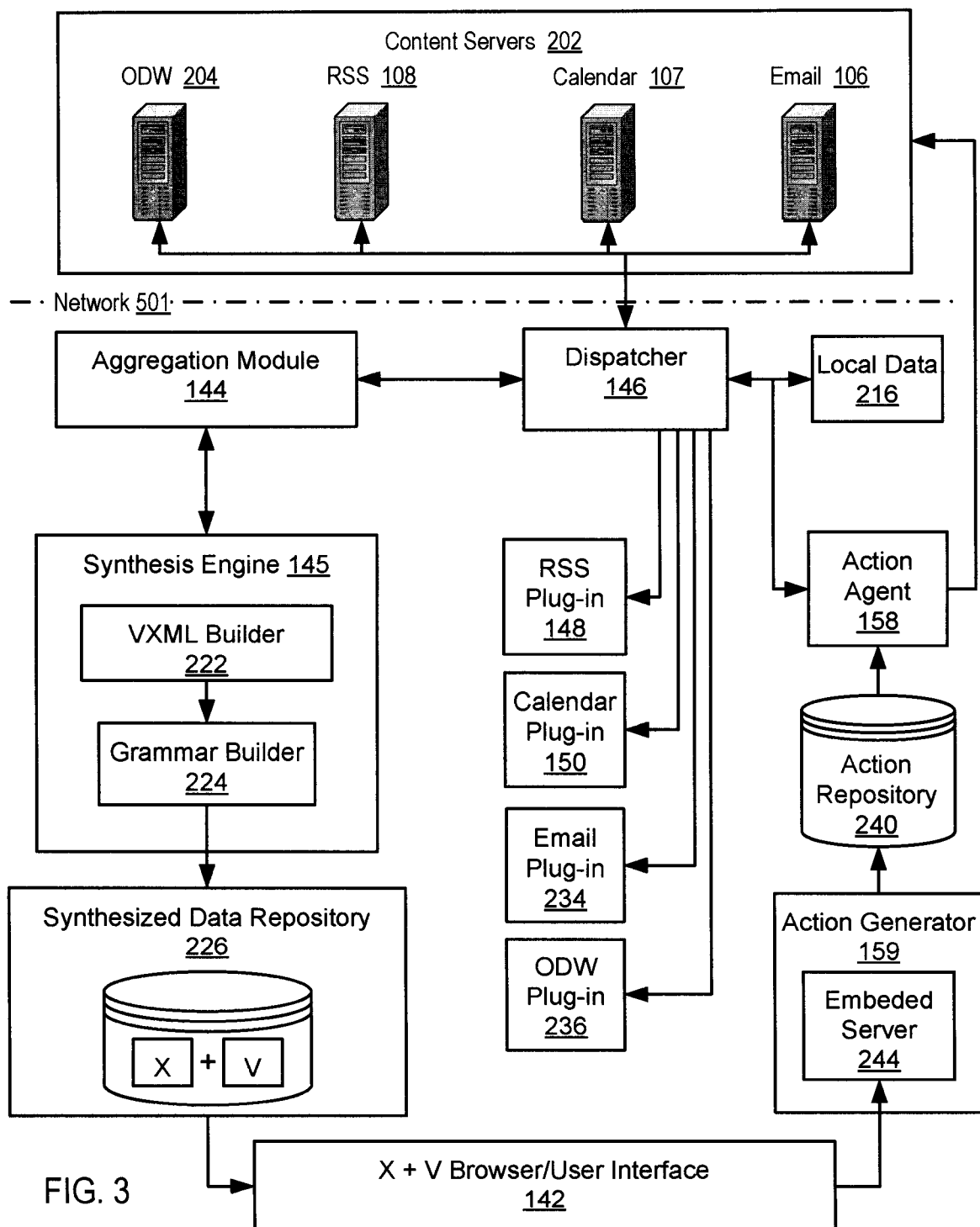


FIG. 2

3/14



4/14

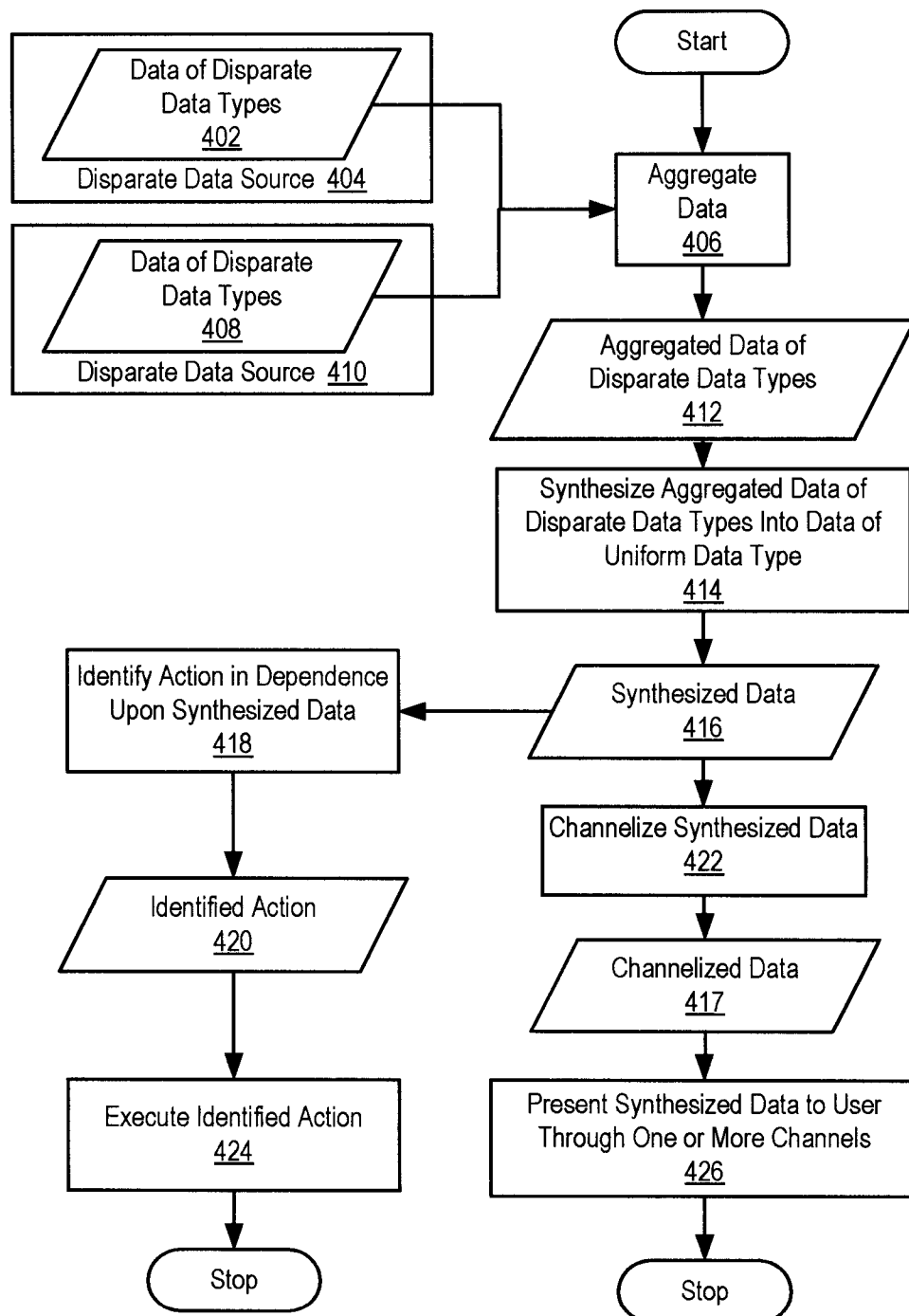


FIG. 4

5/14

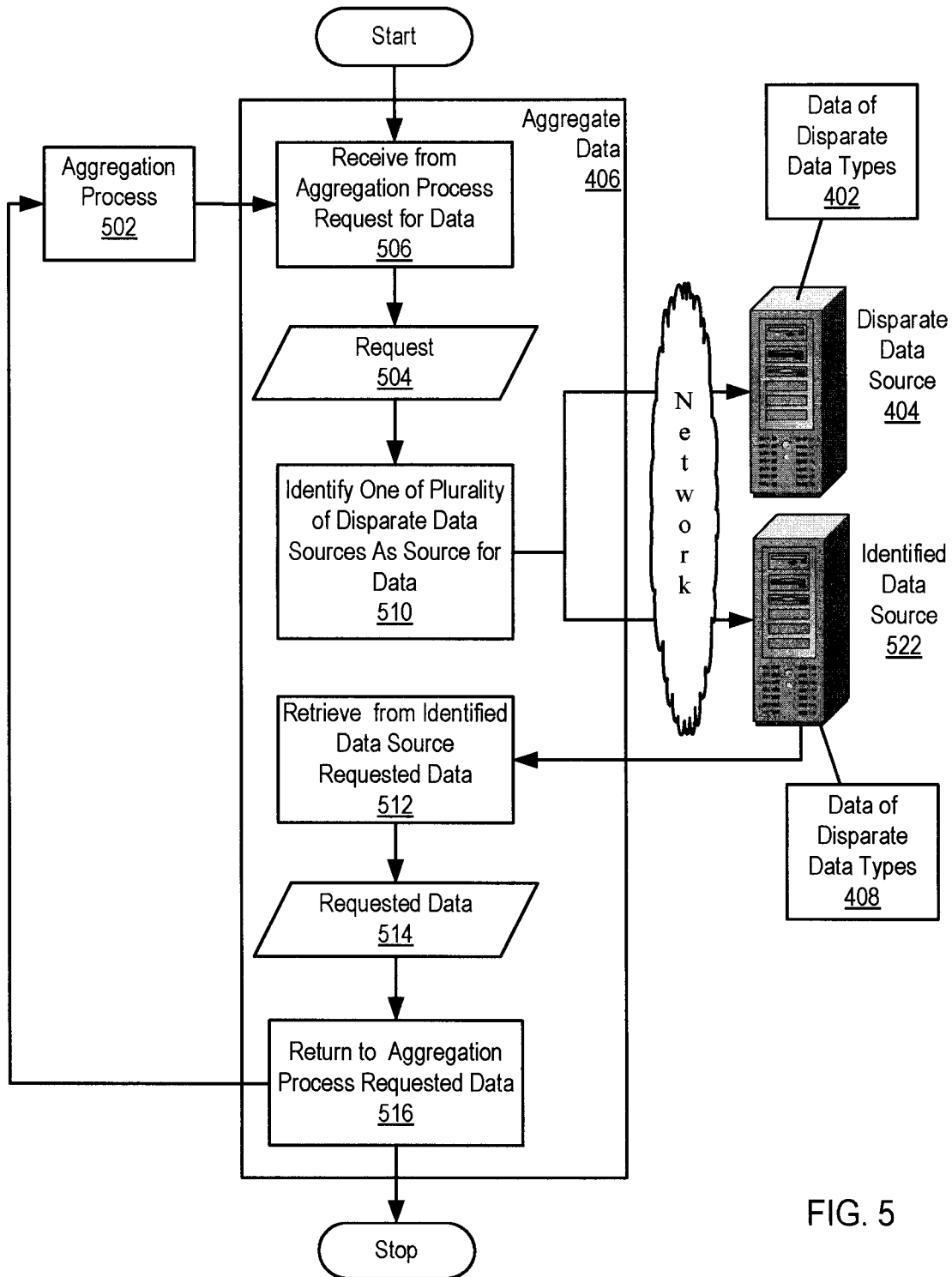
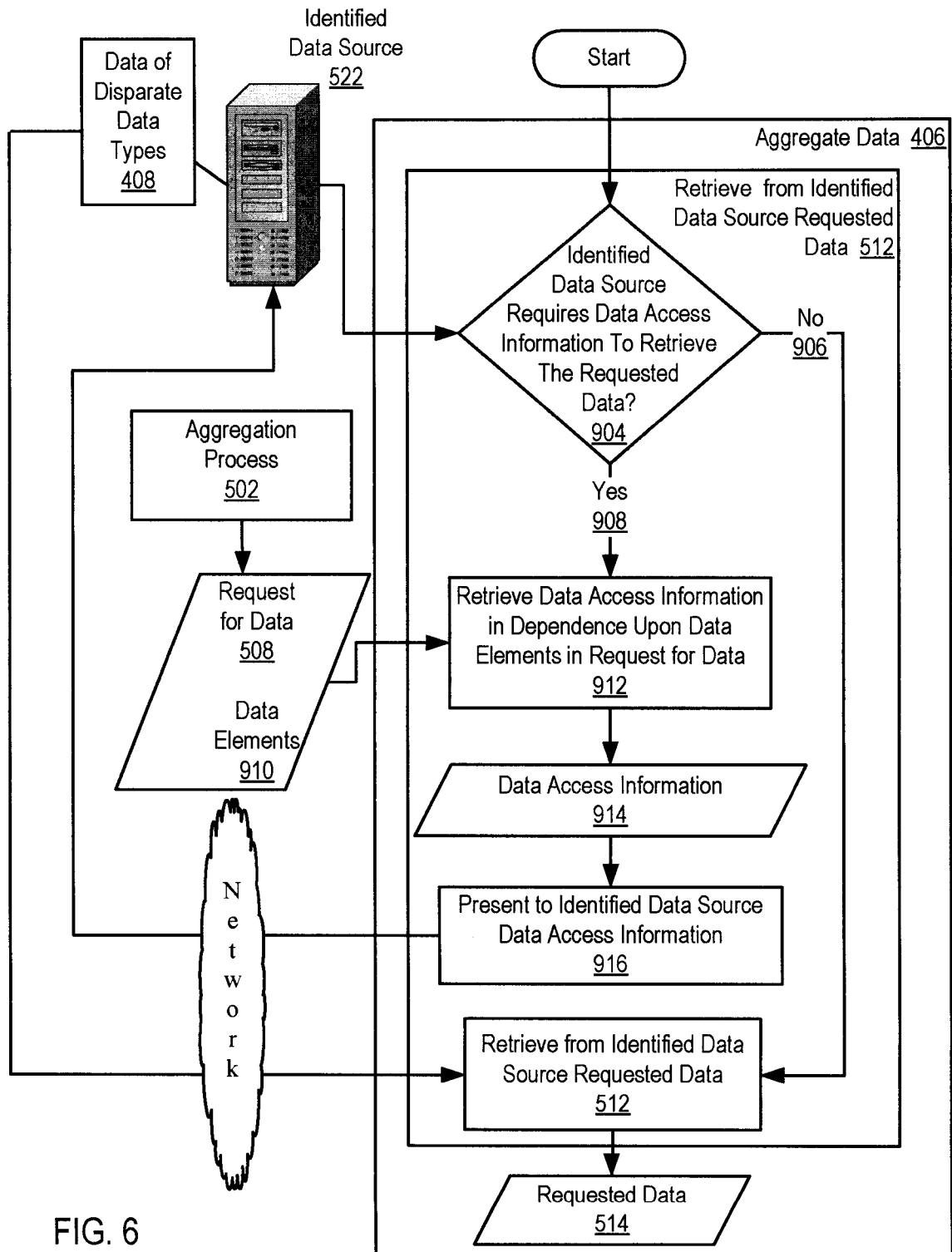
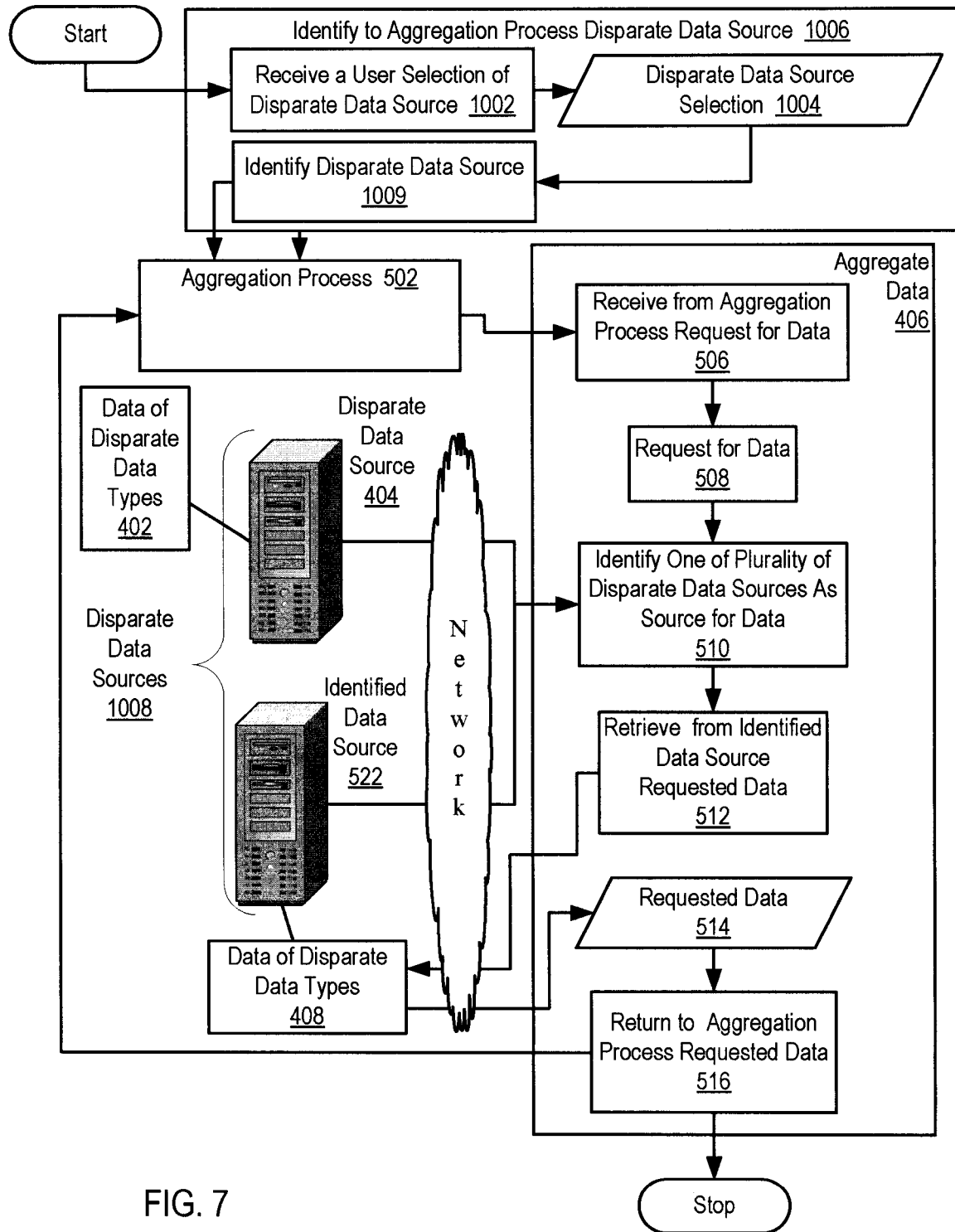


FIG. 5

6/14



7/14



8/14

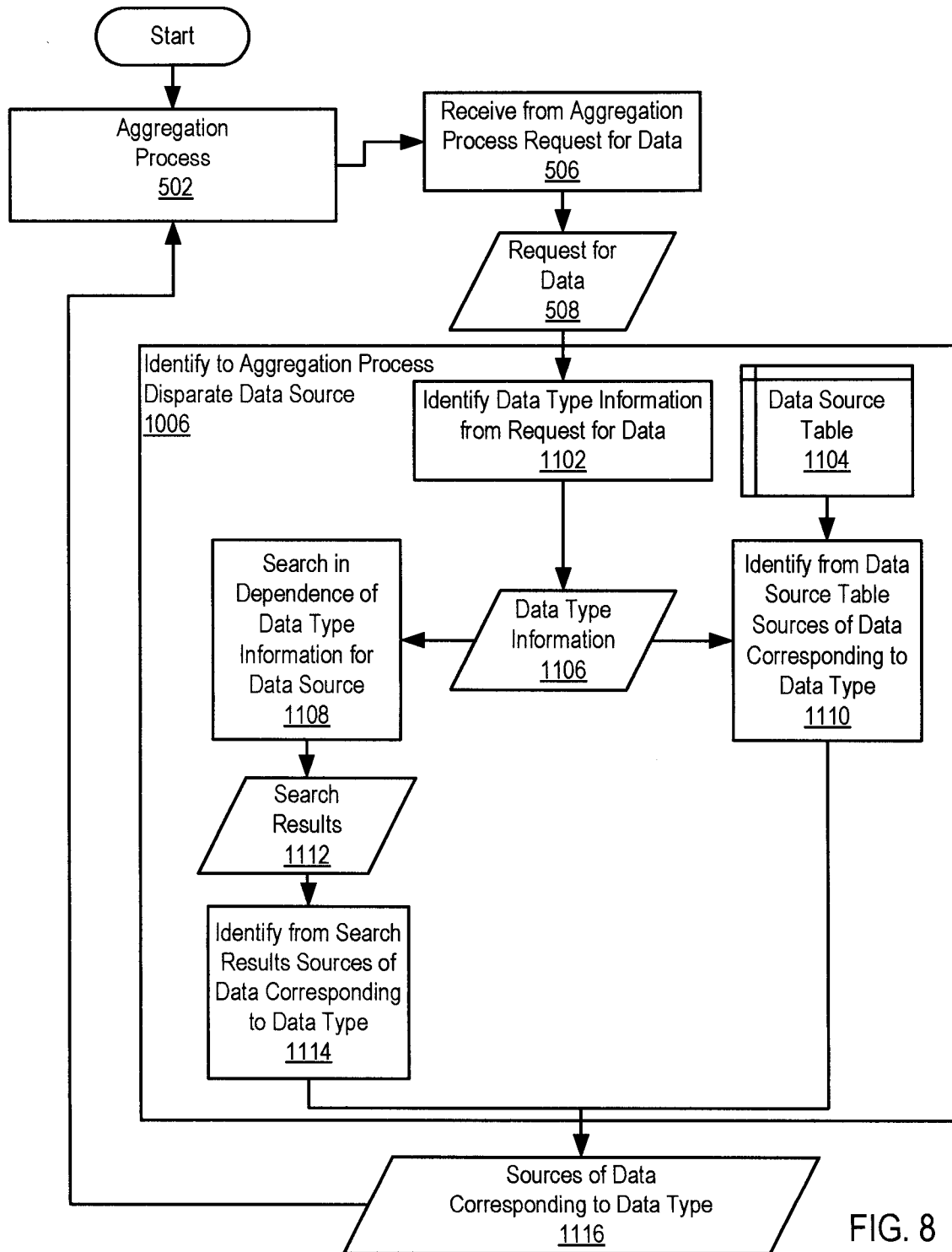
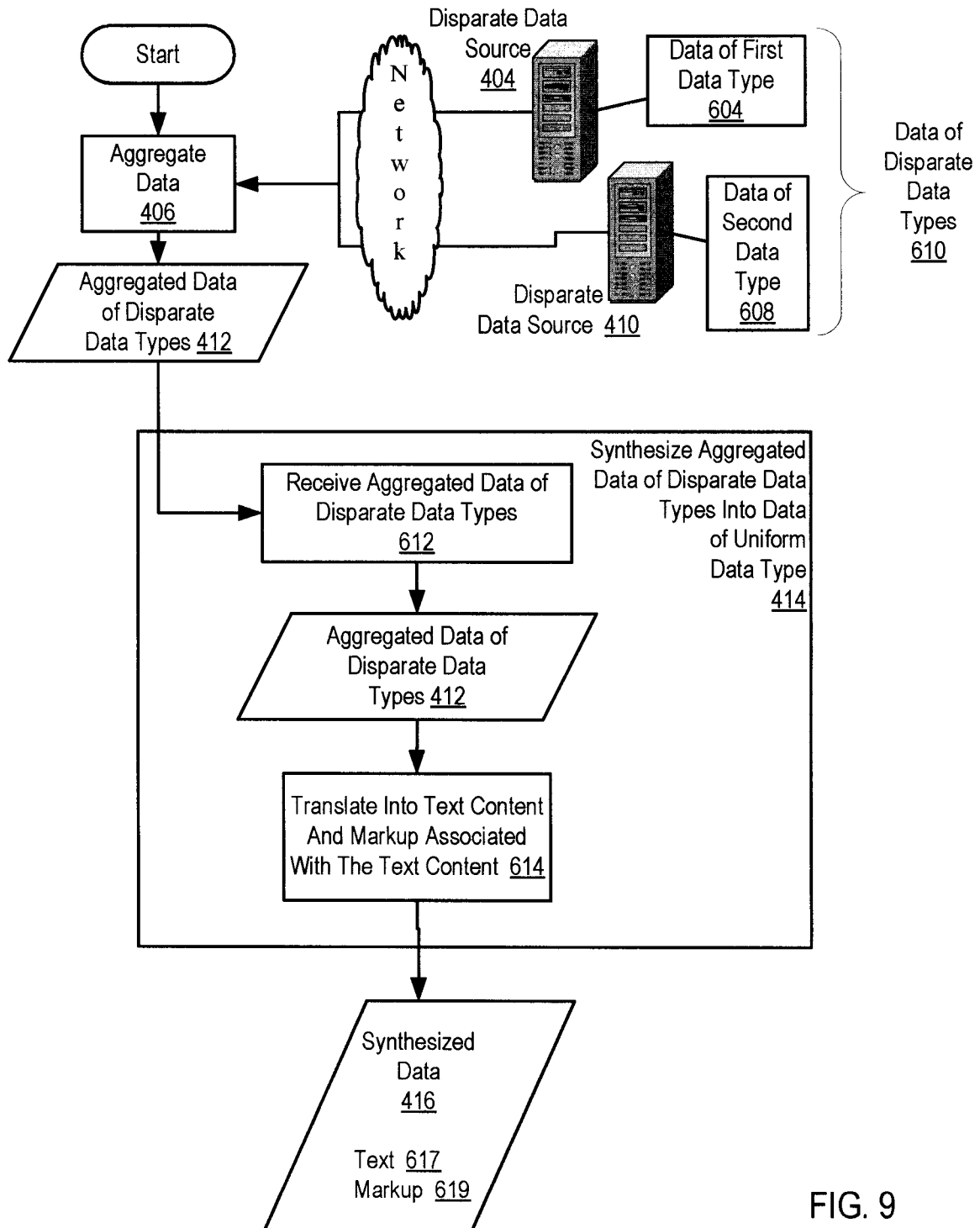


FIG. 8

9/14



10/14

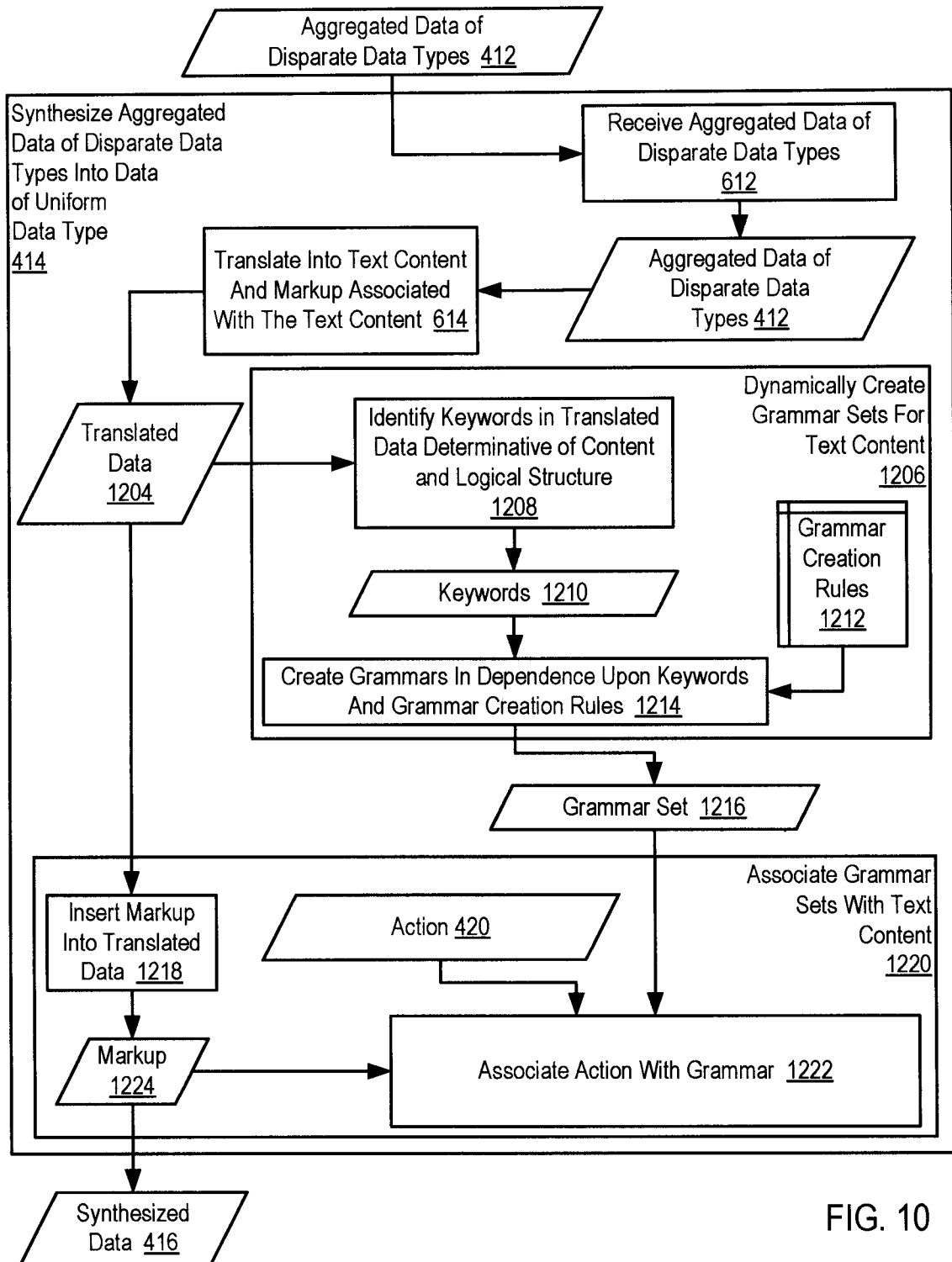
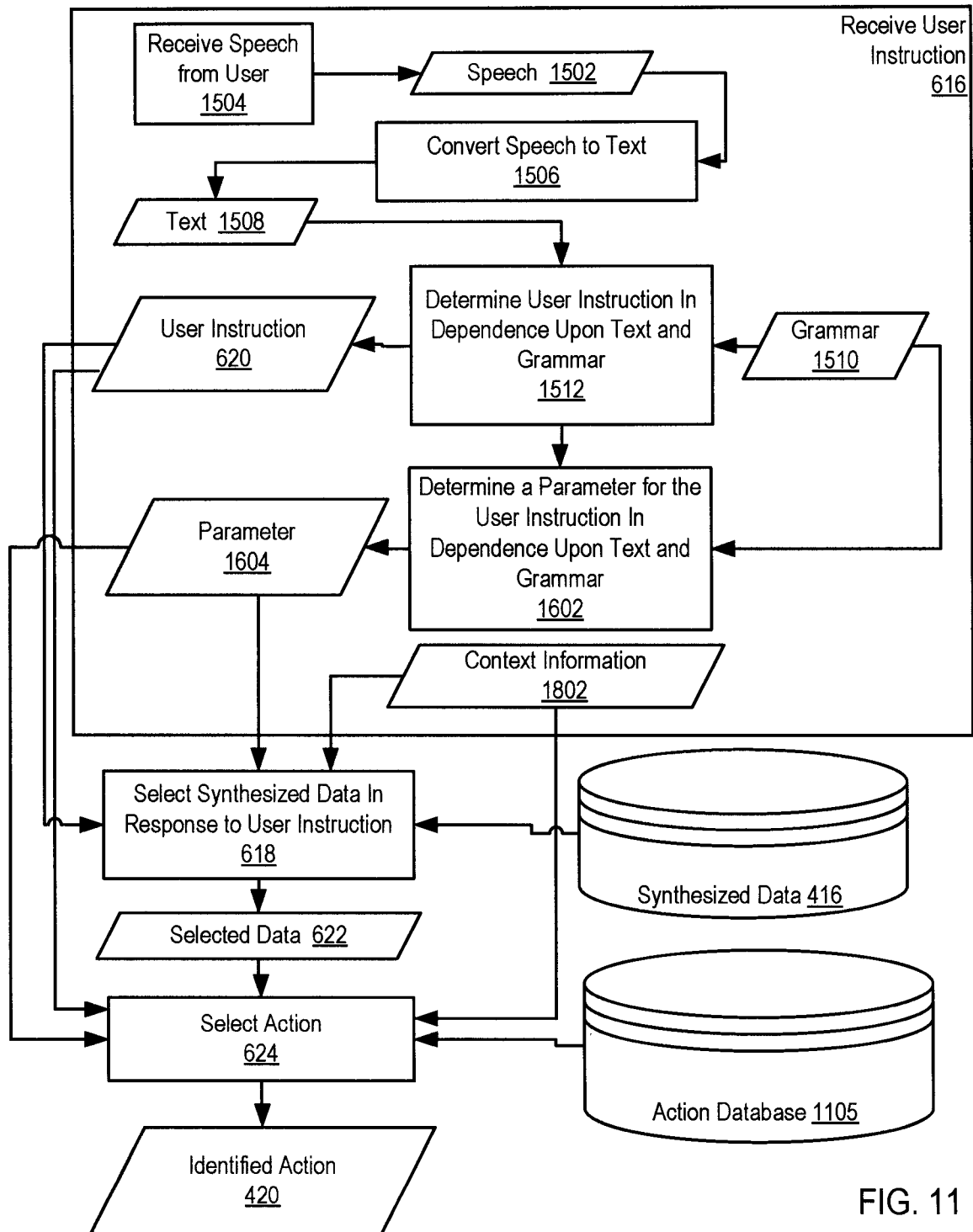


FIG. 10

11/14



12/14

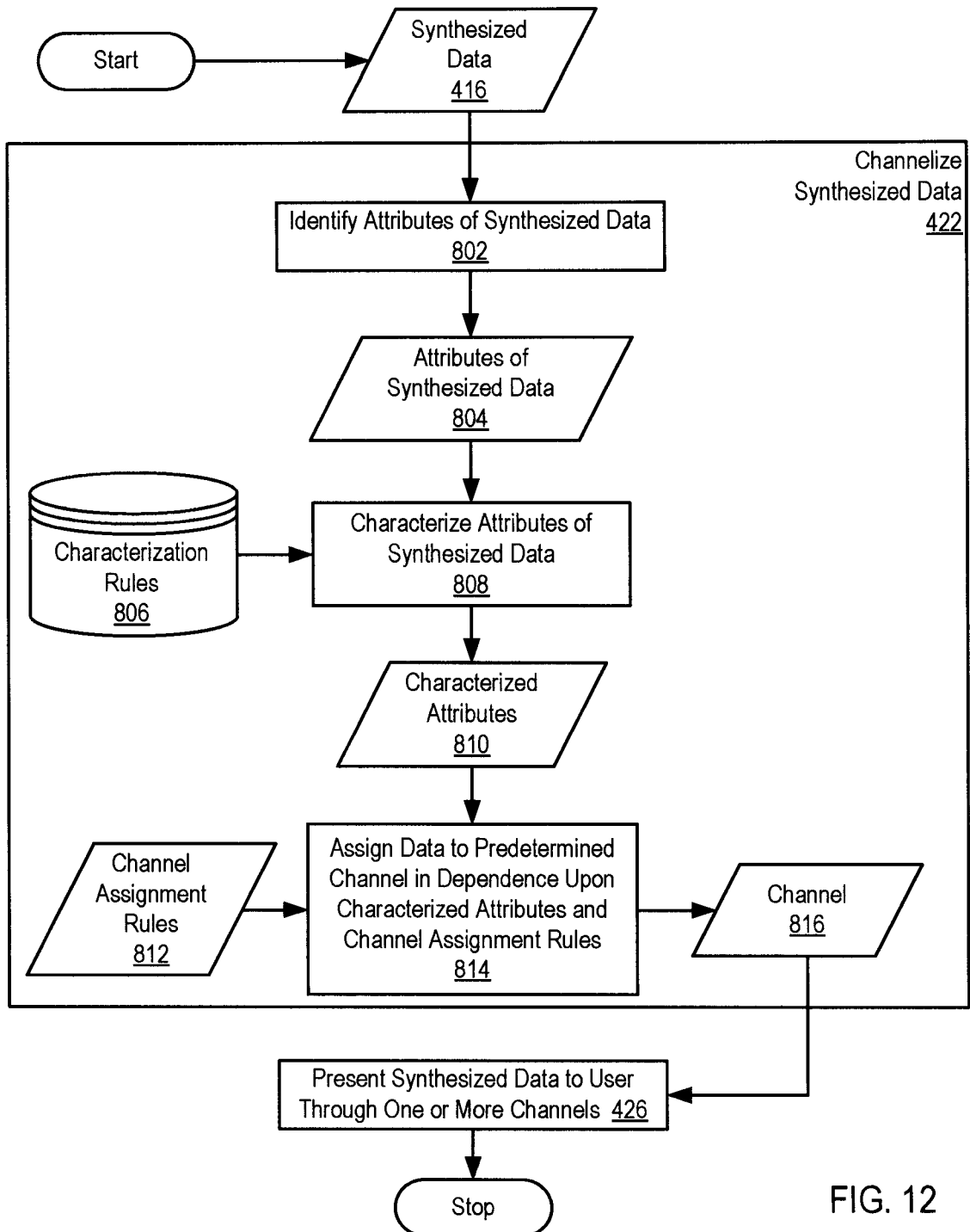
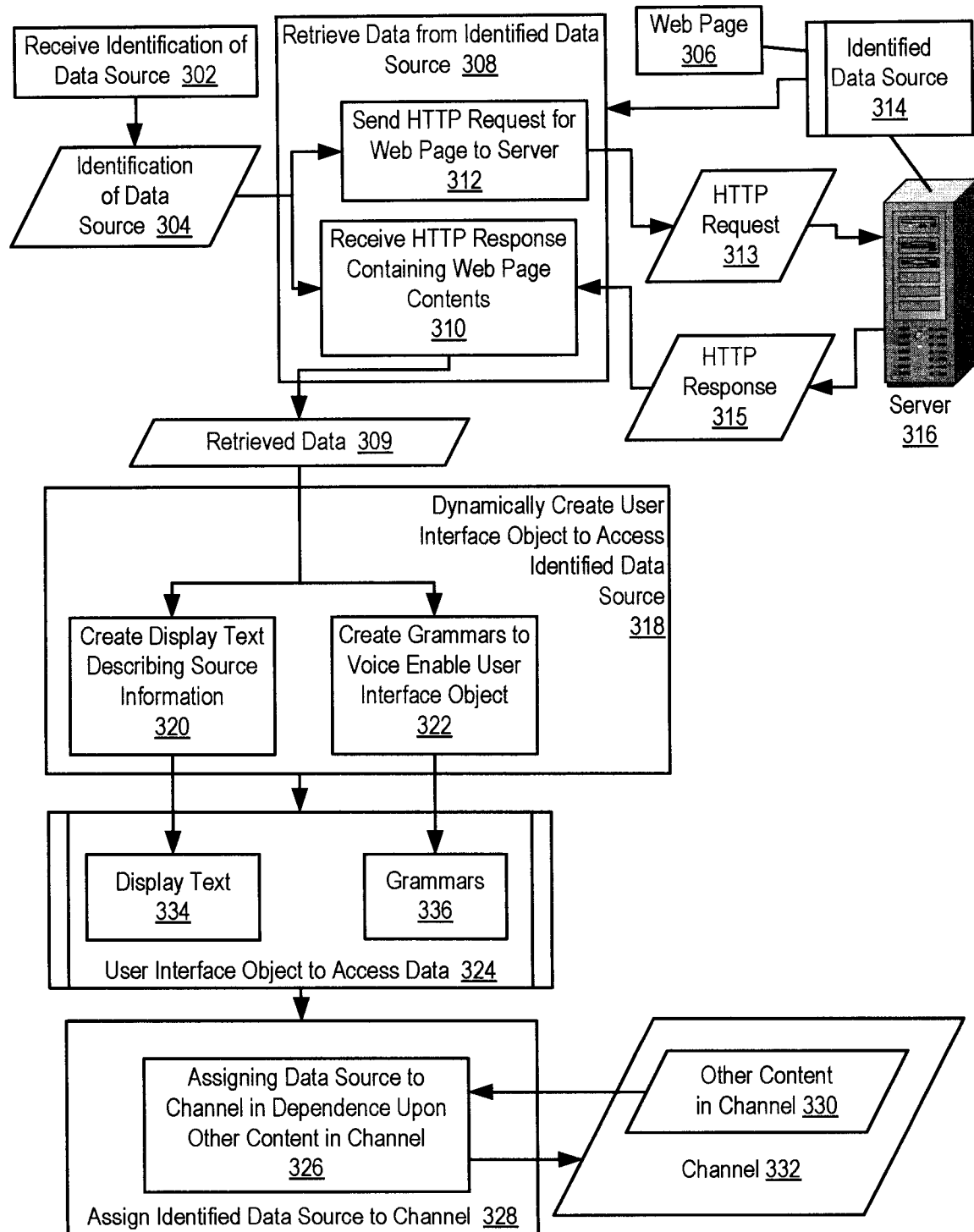


FIG. 12

13/14



14/14

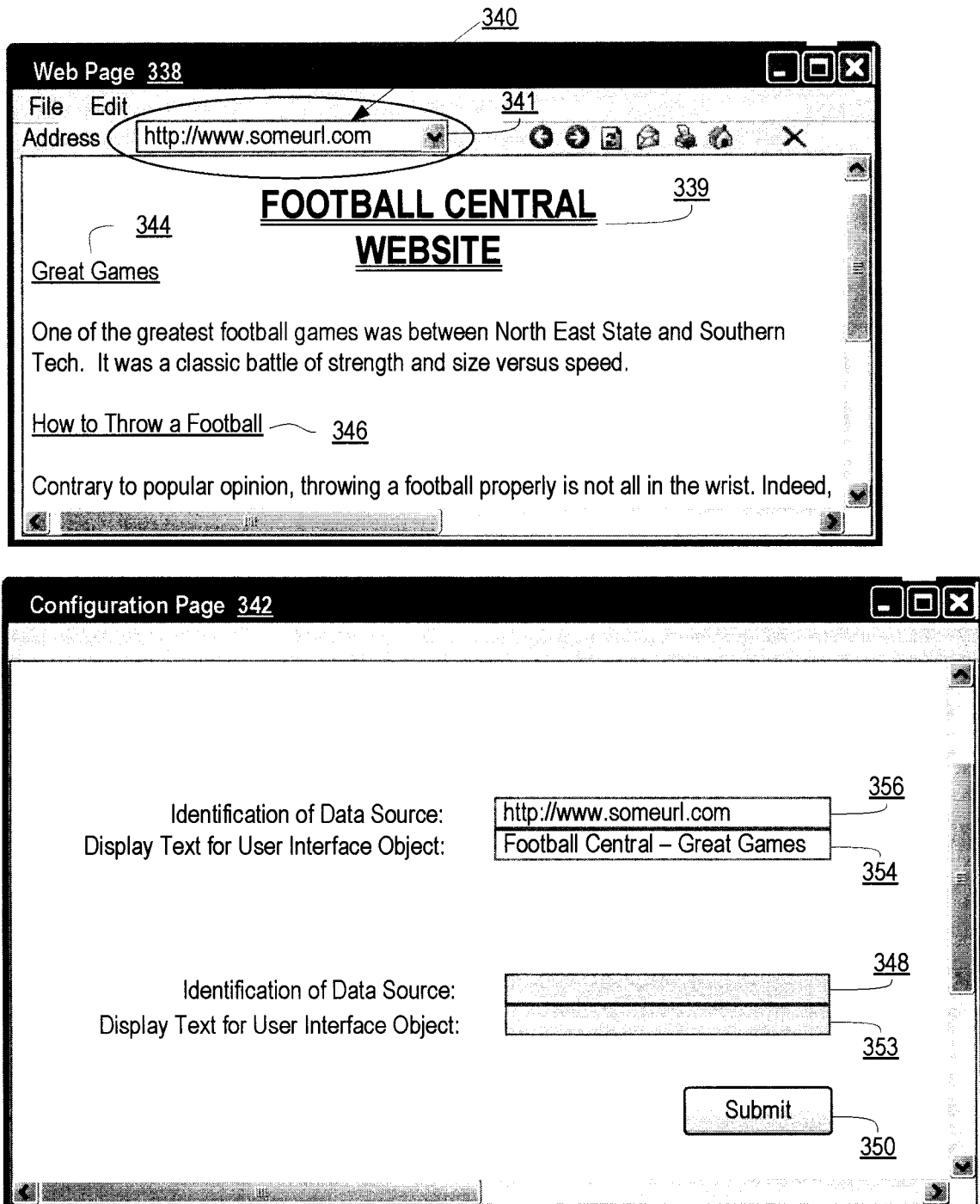


FIG. 14