US 20100057885A1

(19) **United States**

(12) **Patent Application Publication**

Yuen et al.

(10) **Pub. No.: US 2010/0057885 A1**

(43) **Pub. Date: Mar. 4, 2010**

(54) **METHOD FOR AUTOMATED HARVESTING OF DATA FROM A WEB SITE USING A VOICE PORTAL SYSTEM**

(75) Inventors: **Michael S. Yuen**, San Francisco, CA (US); **Leo Chiu**, Daly City, CA (US)

Correspondence Address:
Stevens Law Group
1754 Technology Drive, Suite #226
San Jose, CA 95110 (US)

(73) Assignee: **Apptera, Inc.**, San Bruno, CA (US)

(21) Appl. No.: **12/490,177**

(22) Filed: **Jun. 23, 2009**

Related U.S. Application Data

(63) Continuation of application No. 10/190,077, filed on Jul. 2, 2002, which is a continuation-in-part of application No. 10/173,333, filed on Jun. 14, 2002.

(60) Provisional application No. 60/302,736, filed on Jul. 3, 2001.

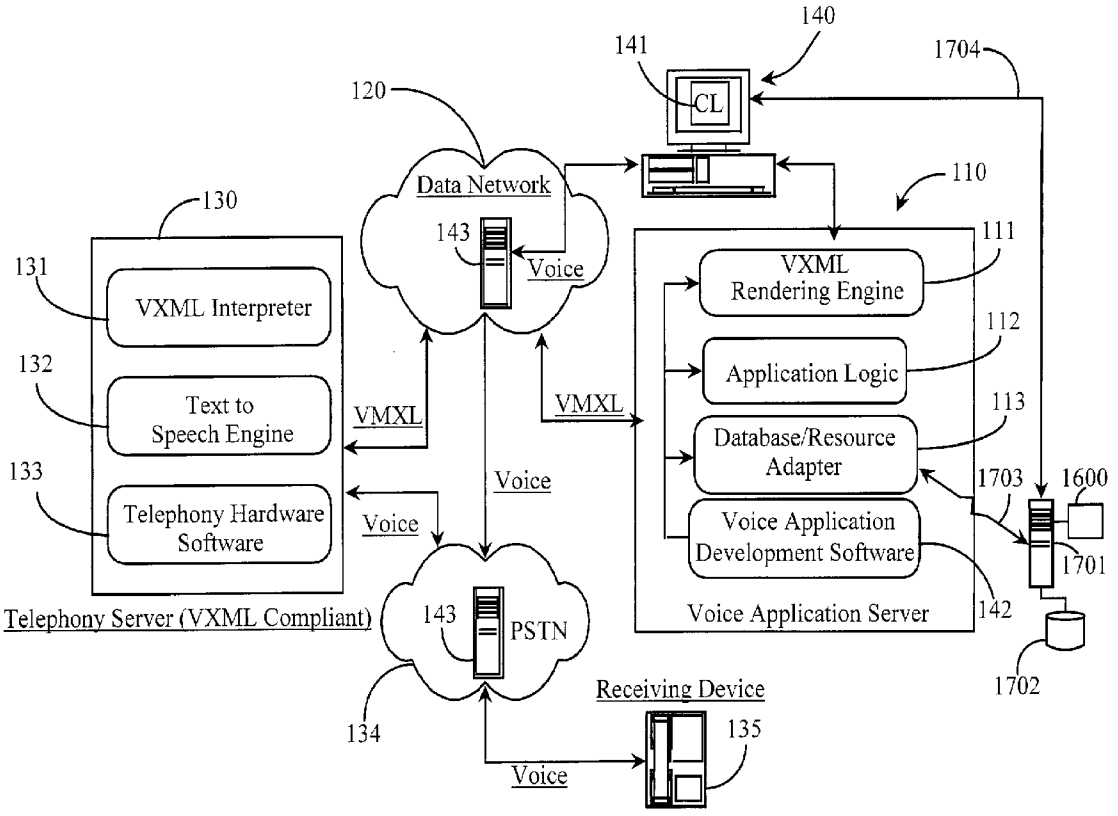
Publication Classification

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/219**

(57) **ABSTRACT**

A system is provided for developing and deploying a voice application using Web-based data as source data over a communications network to one or more recipients. The system has a voice application server capable of accessing a network server and Website hosted therein, a network communications server for routing the voice applications to their intended recipients, and a computer station running voice application software having control access to at least the voice application server, the computer station capable of accessing the network server and Website. An operator of the computer station creates and provides templates for the voice application server to use in data-to-voice rendering.



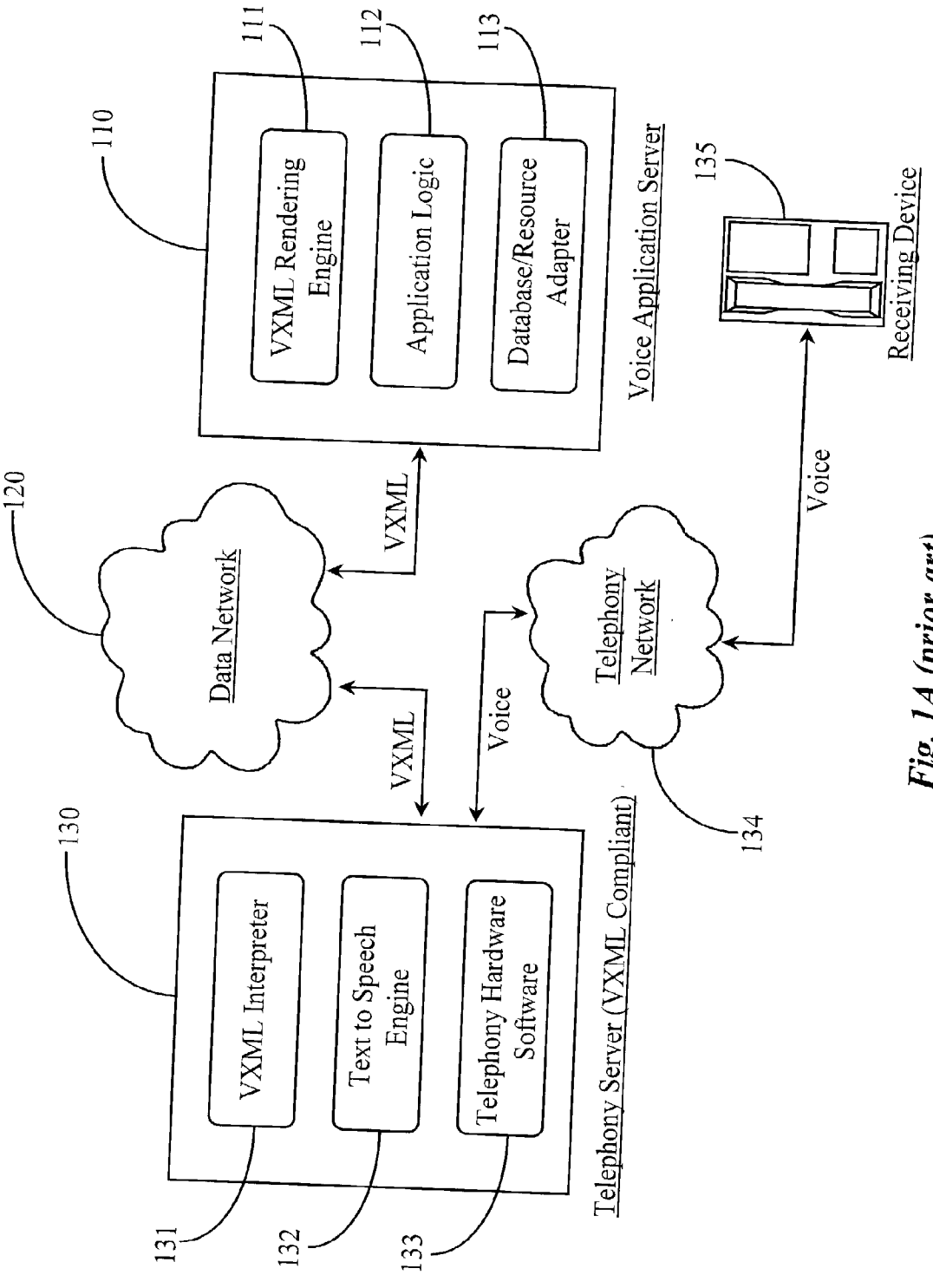


Fig. 1A (prior art)

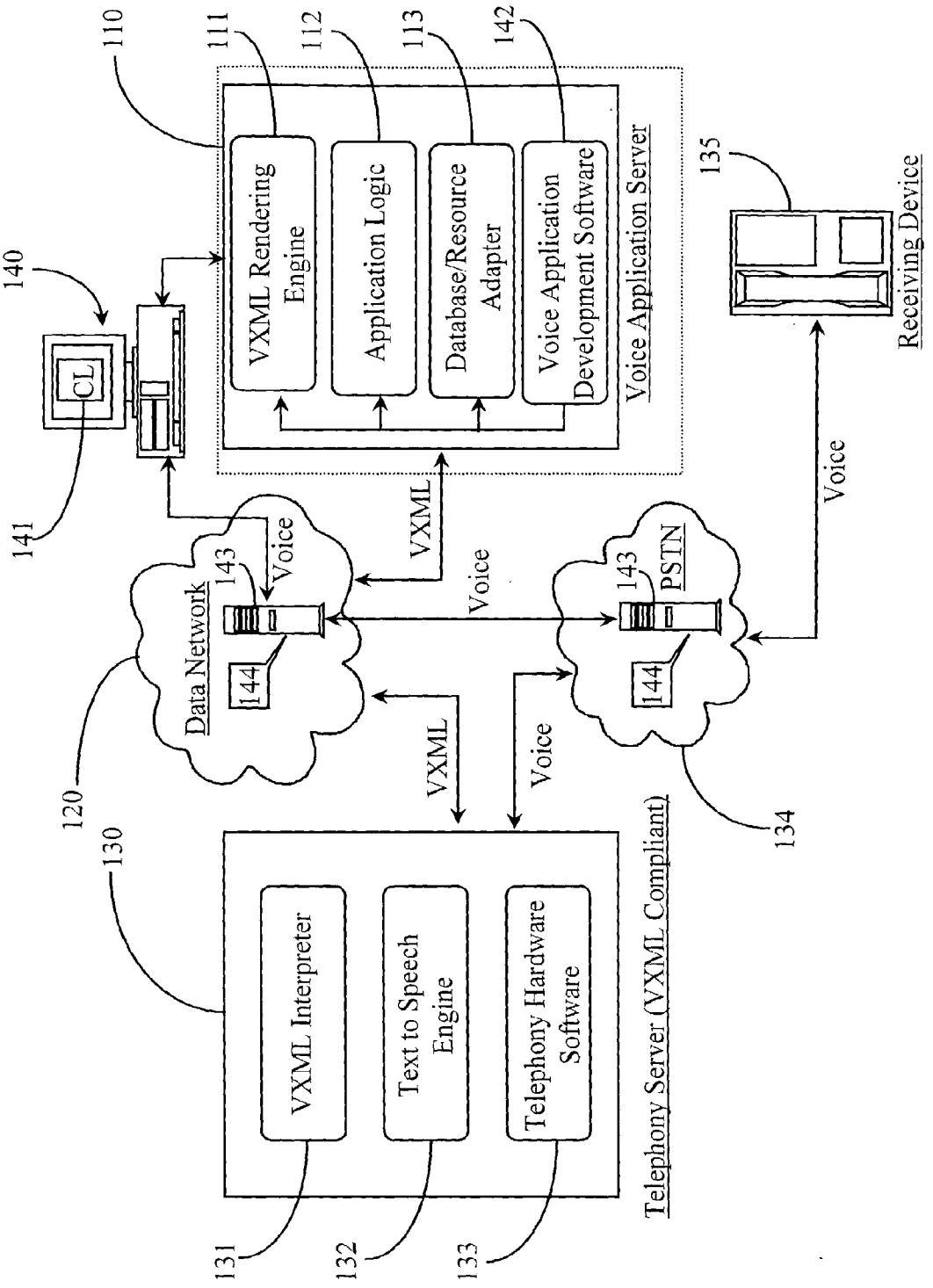

110
111
112
113
142
VXML Rendering Engine
Application Logic
Database/Resource Adapter
Voice Application Development Software
Voice Application Server
140
CL
141
135
Receiving Device
Voice
VXML
Voice
Data Network
143
144
120
PSTN
143
144
134
VXML
Voice
130
VXML Interpreter
Text to Speech Engine
Telephony Hardware Software
Telephony Server (VXML Compliant)
131
132
133

Fig. 1B

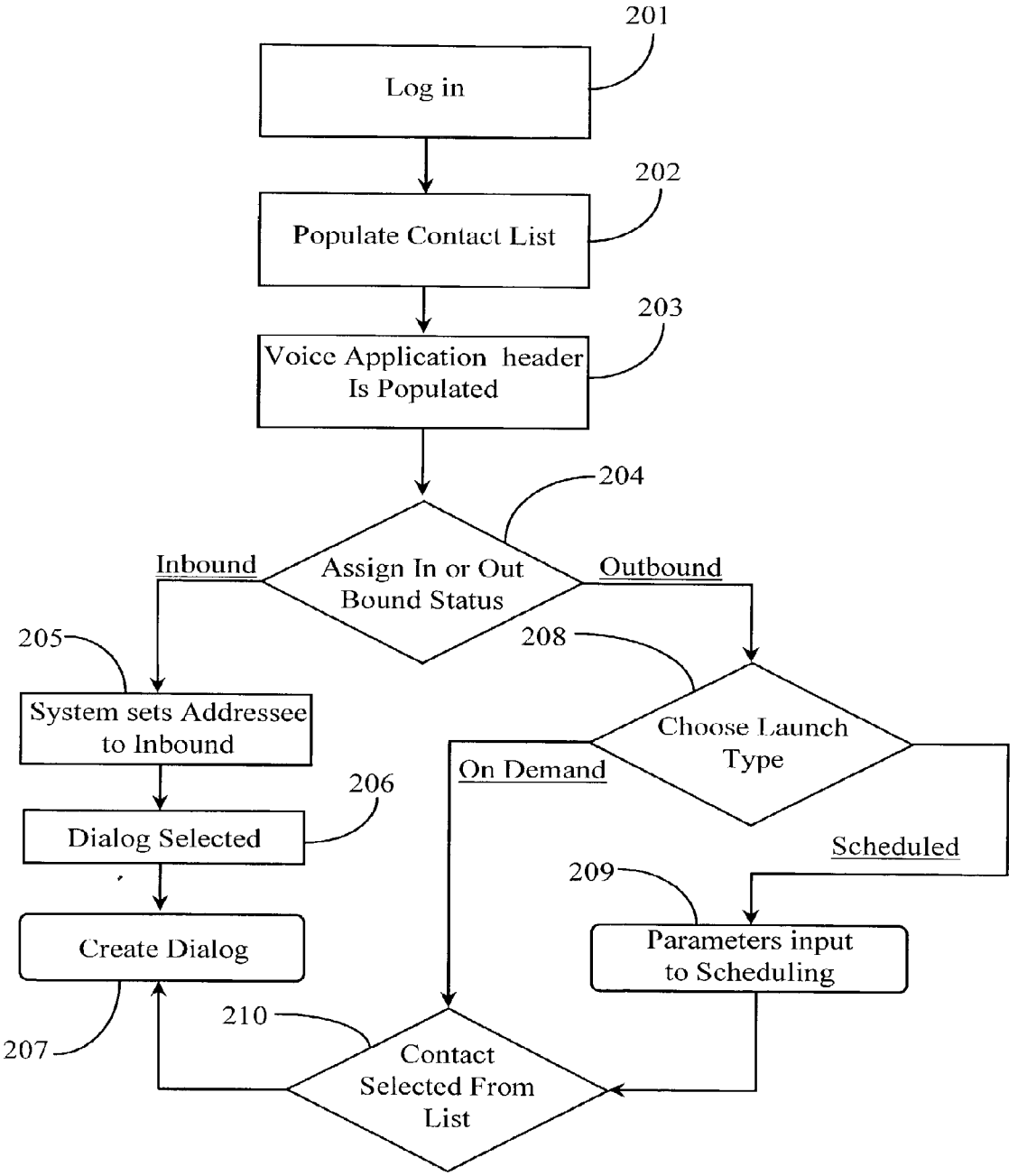

201
Log in
202
Populate Contact List
203
Voice Application header Is Populated
204
Assign In or Out Bound Status
Inbound
Outbound
205
System sets Addressee to Inbound
206
Dialog Selected
Create Dialog
207
208
Choose Launch Type
On Demand
Scheduled
209
Parameters input to Scheduling
210
Contact Selected From List

Fig. 2

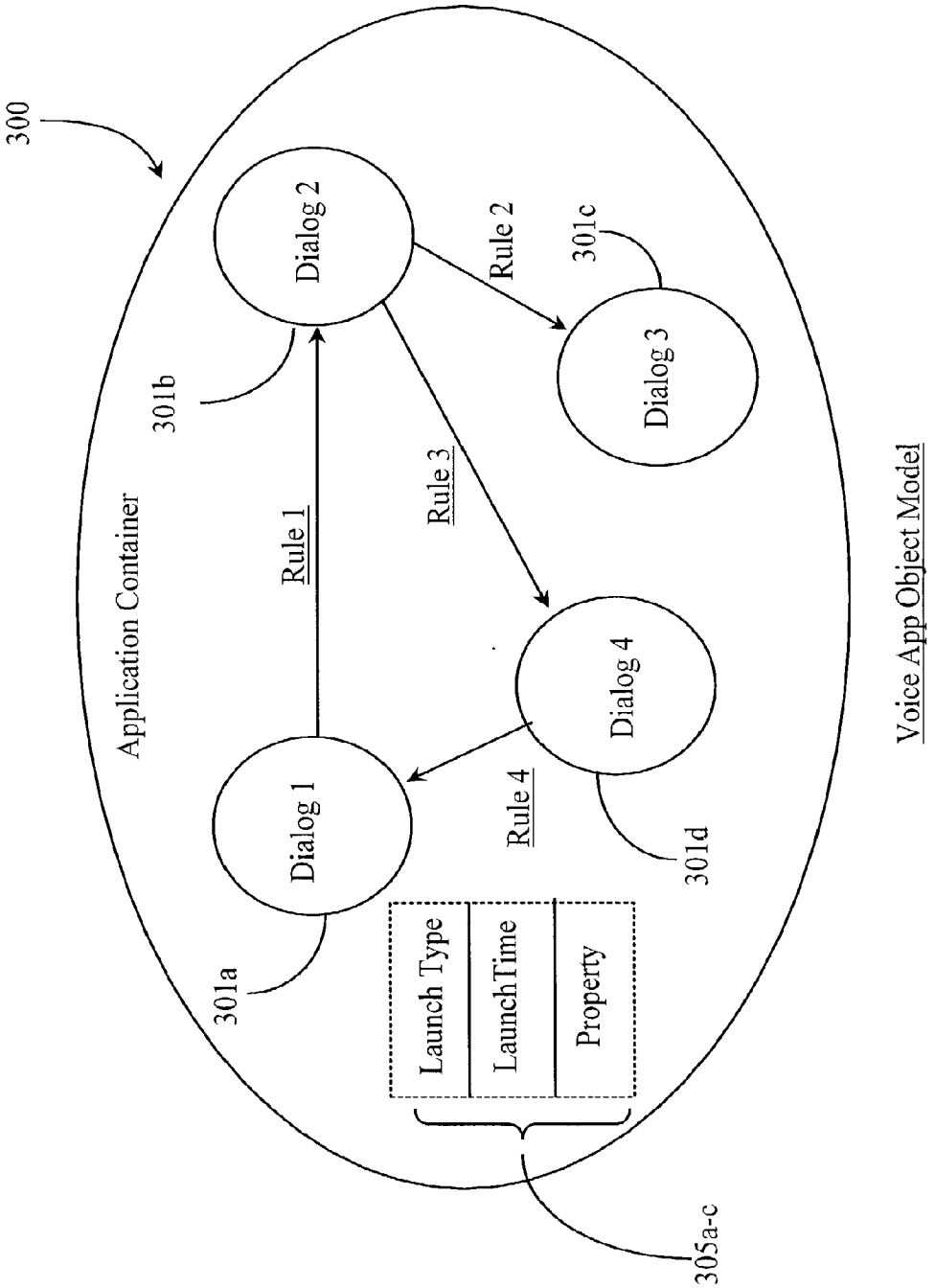

300
Application Container
Dialog 1
Dialog 2
Dialog 3
Dialog 4
Rule 1
Rule 2
Rule 3
Rule 4
301a
301b
301c
301d
Launch Type
LaunchTime
Property
305a-c
Voice App Object Model

Fig. 3

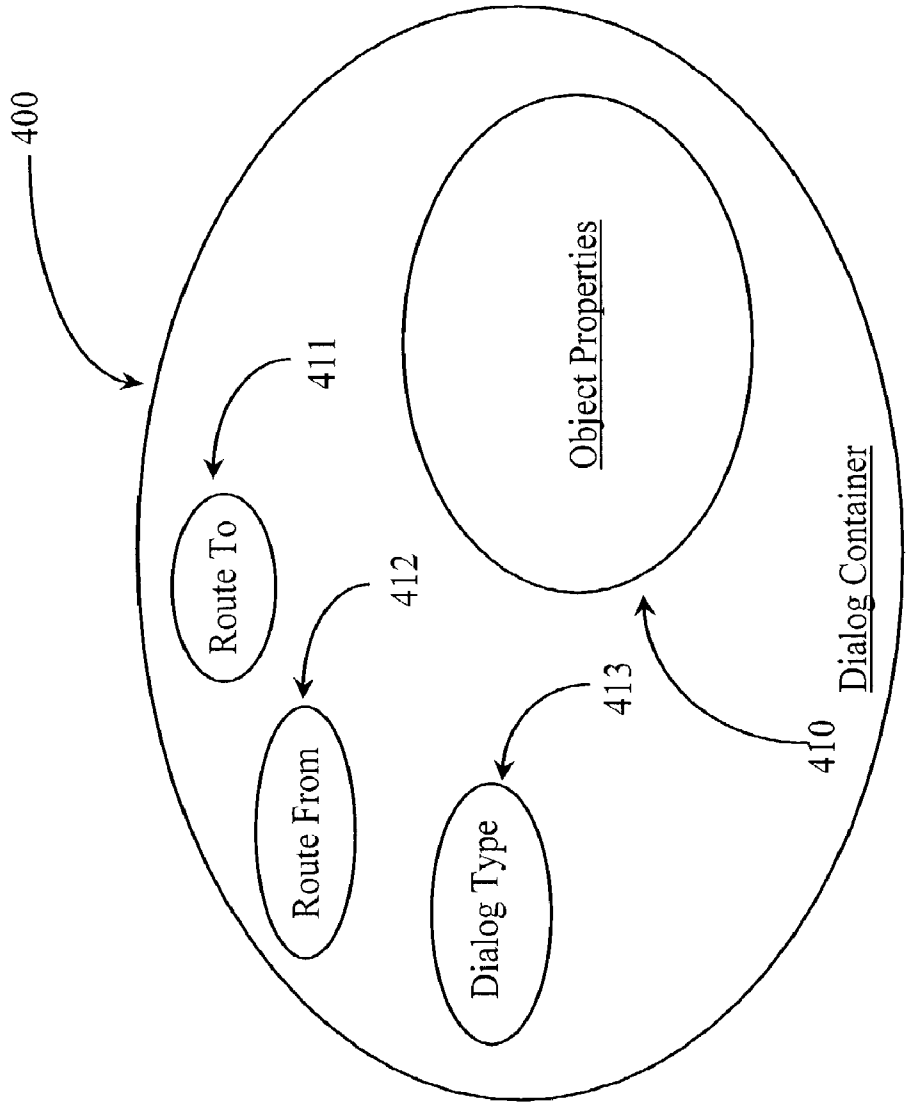

400
411
412
413
410
Route To
Route From
Dialog Type
Object Properties
Dialog Container
Dialog Object Model

Fig. 4

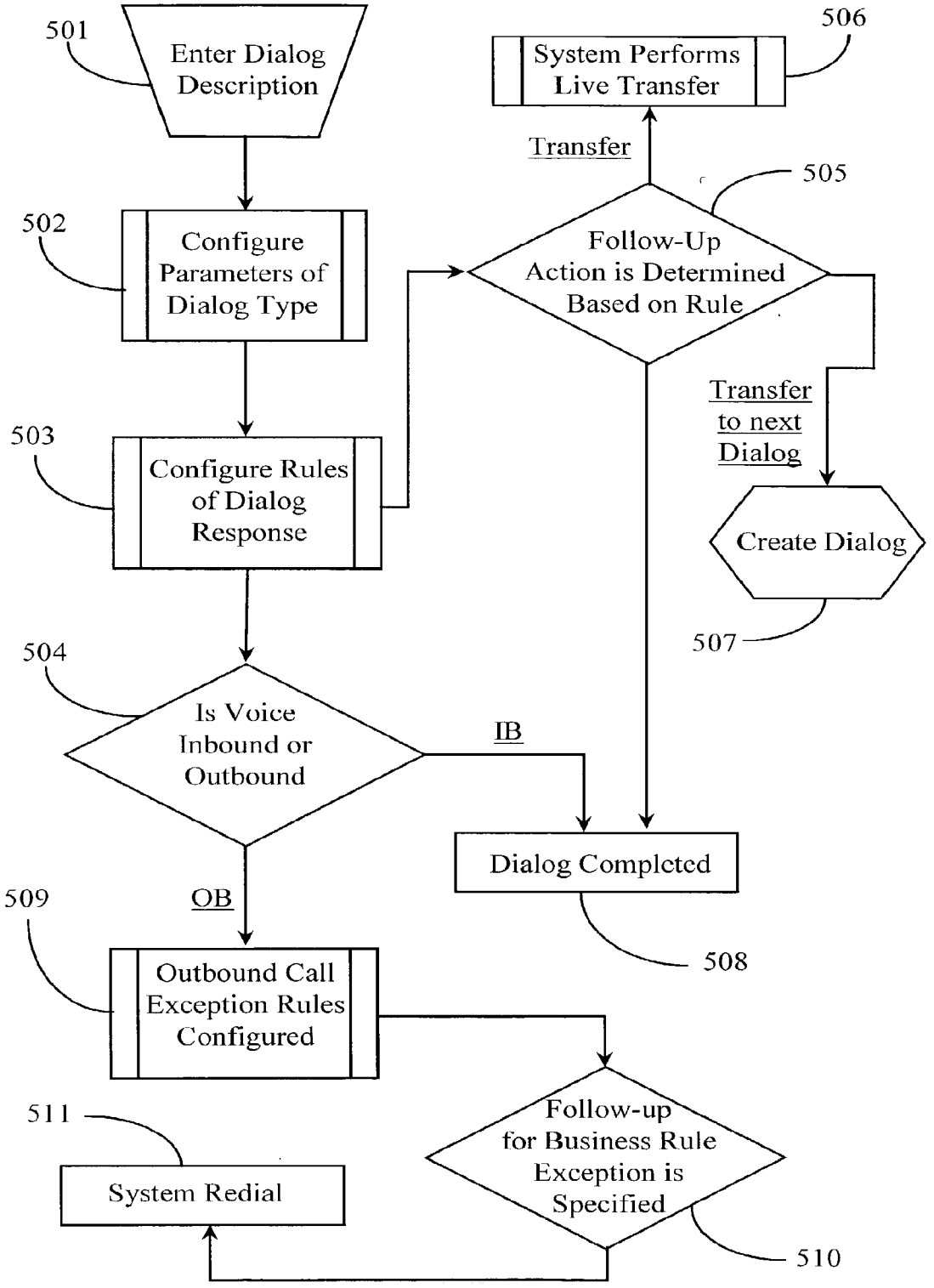

501
Enter Dialog Description
502
Configure Parameters of Dialog Type
503
Configure Rules of Dialog Response
504
Is Voice Inbound or Outbound
IB
OB
509
Outbound Call Exception Rules Configured
511
System Redial
Follow-up for Business Rule Exception is Specified
510
Dialog Completed
508
Follow-Up Action is Determined Based on Rule
505
Transfer
System Performs Live Transfer
506
Transfer to next Dialog
Create Dialog
507

Fig. 5

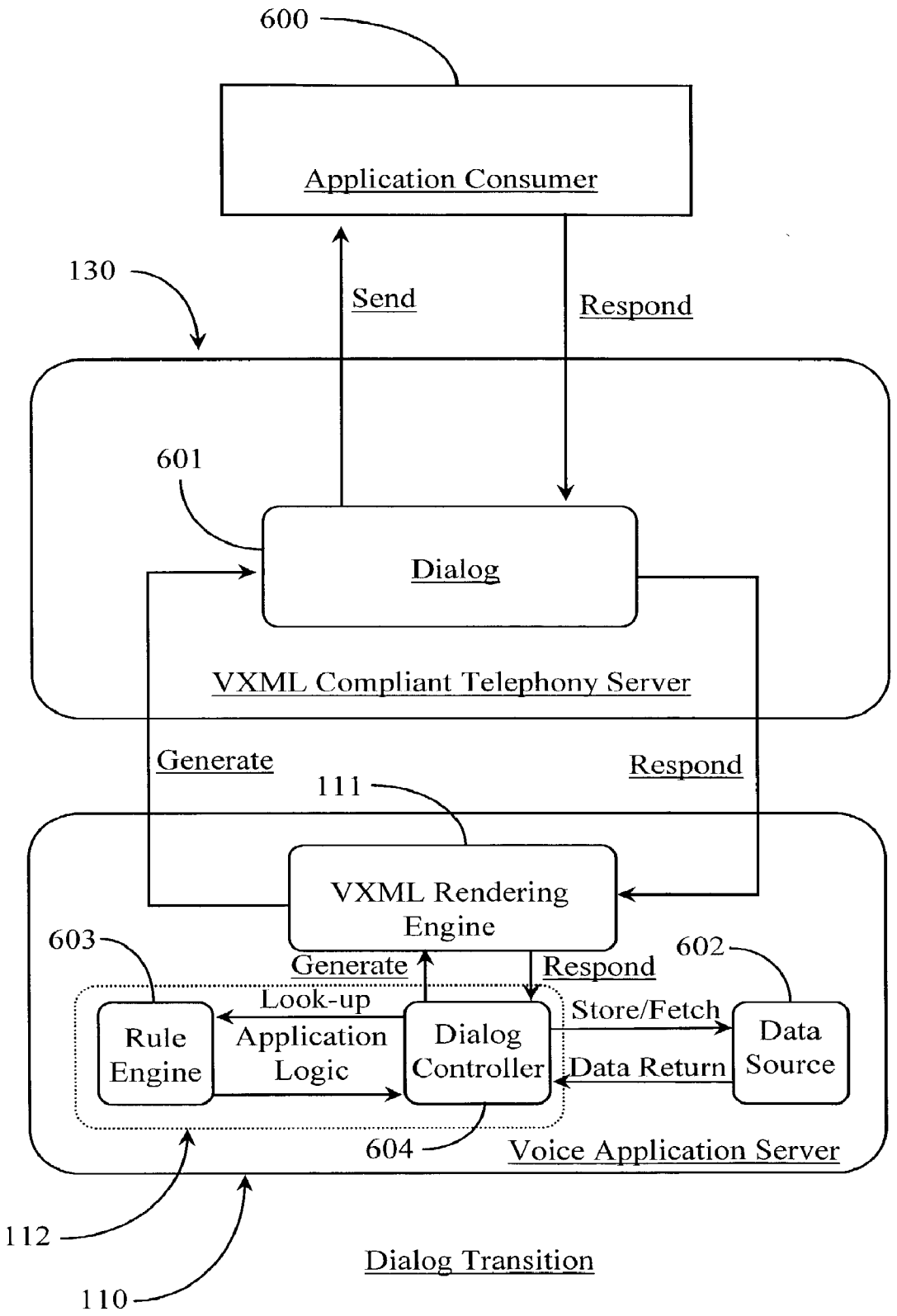

600
Application Consumer
130
Send
Respond
601
Dialog
VXML Compliant Telephony Server
Generate
Respond
111
VXML Rendering Engine
603
602
Generate
Respond
Look-up Application Logic
Store/Fetch
Data Return
Rule Engine
Dialog Controller
Data Source
604
Voice Application Server
112
110
Dialog Transition

Fig. 6

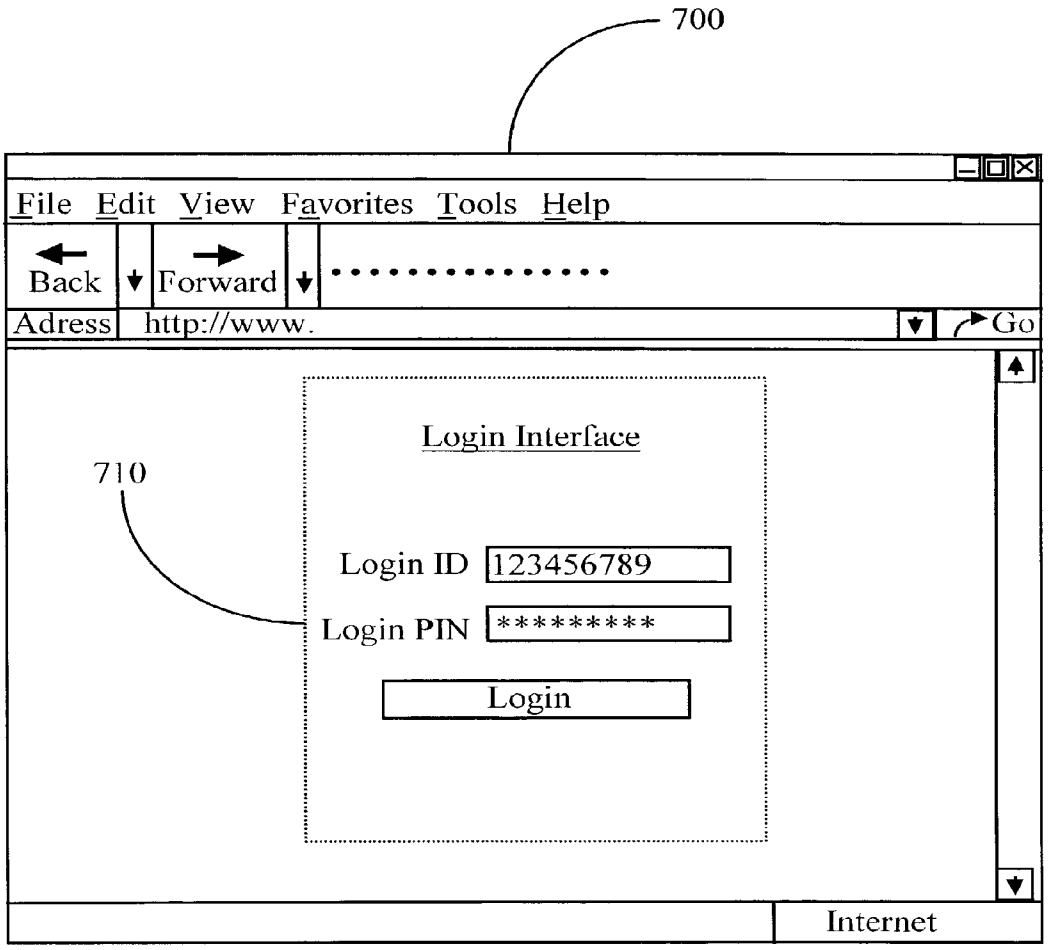

700
File Edit View Favorites Tools Help
Back
Forward
Adress
http://www.
Go
Login Interface
710
Login ID
123456789
Login PIN
****
Login
Internet

Fig. 7

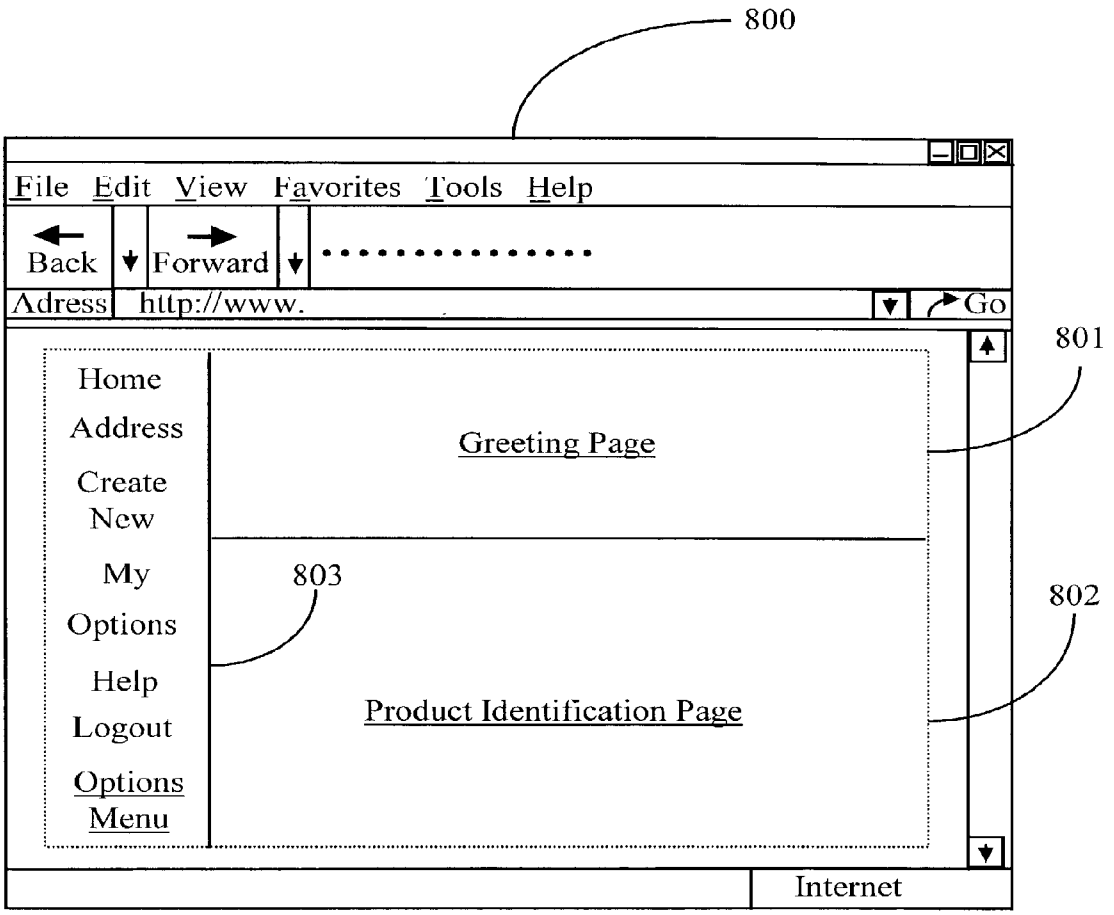

800
File Edit View Favorites Tools Help
Back
Forward
Adress
http://www.
Go
801
Home
Address
Create New
My Options
Help
Logout
Options Menu
Greeting Page
803
802
Product Identification Page
Internet

Fig. 8

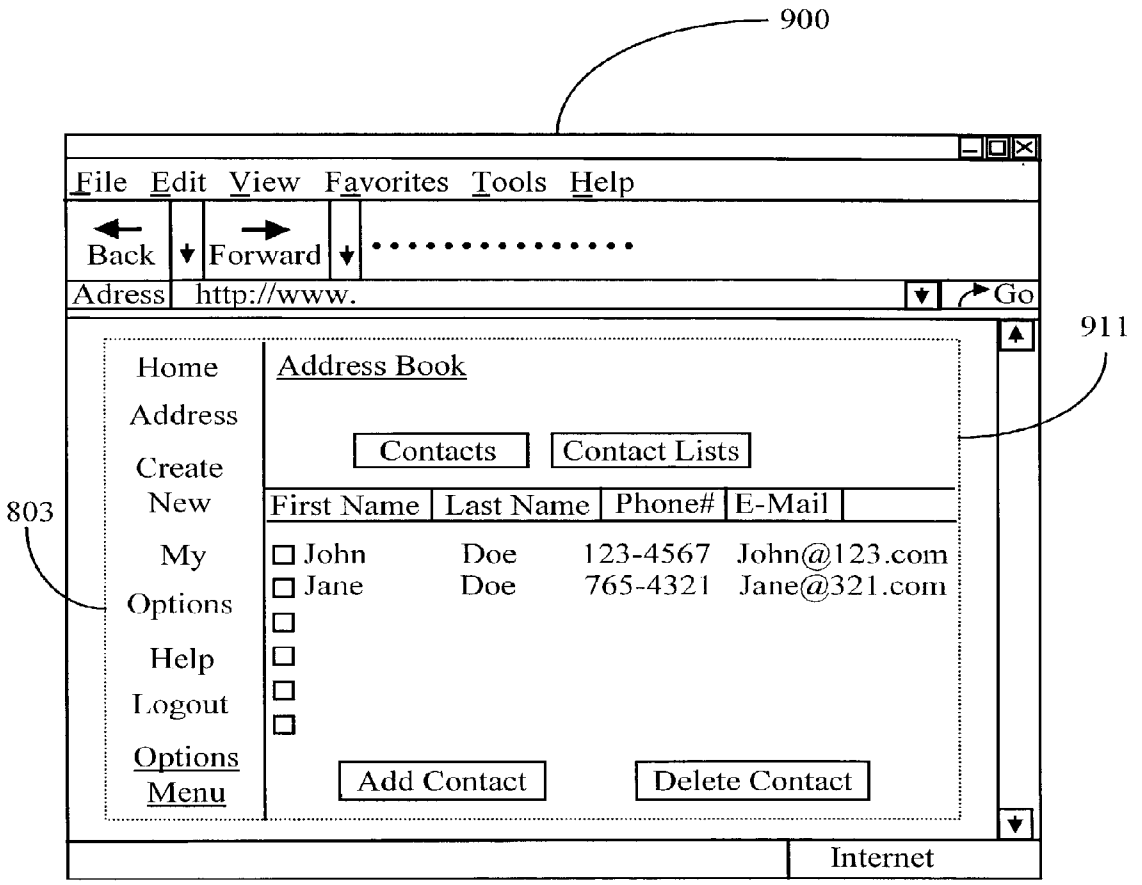

900
File Edit View Favorites Tools Help
Back
Forward
Adress
http://www.
Go
911
803
Home
Address
Create New
My Options
Help
Logout
Options Menu
Address Book
Contacts
Contact Lists
First Name
Last Name
Phone#
E-Mail
John Doe 123-4567 John@123.com
Jane Doe 765-4321 Jane@321.com
Add Contact
Delete Contact
Internet

Fig. 9

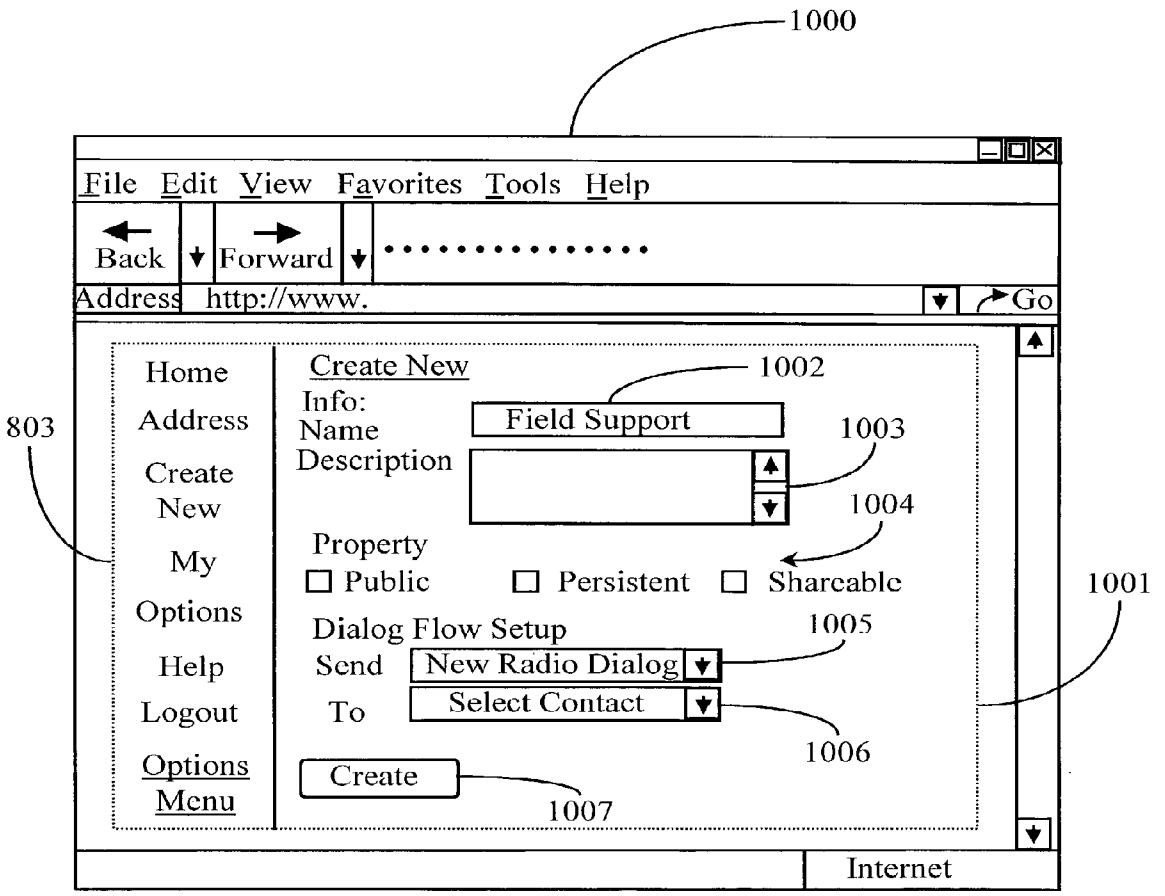

1000
File Edit View Favorites Tools Help
Back
Forward
Address http://www.
Go
803
Home
Address
Create New
My
Options
Help
Logout
Options Menu
Create New
1002
Info:
Name
Field Support
1003
Description
1004
Property
Public
Persistent
Shareable
1001
Dialog Flow Setup
1005
Send
New Radio Dialog
To
Select Contact
1006
Create
1007
Internet

Fig. 10

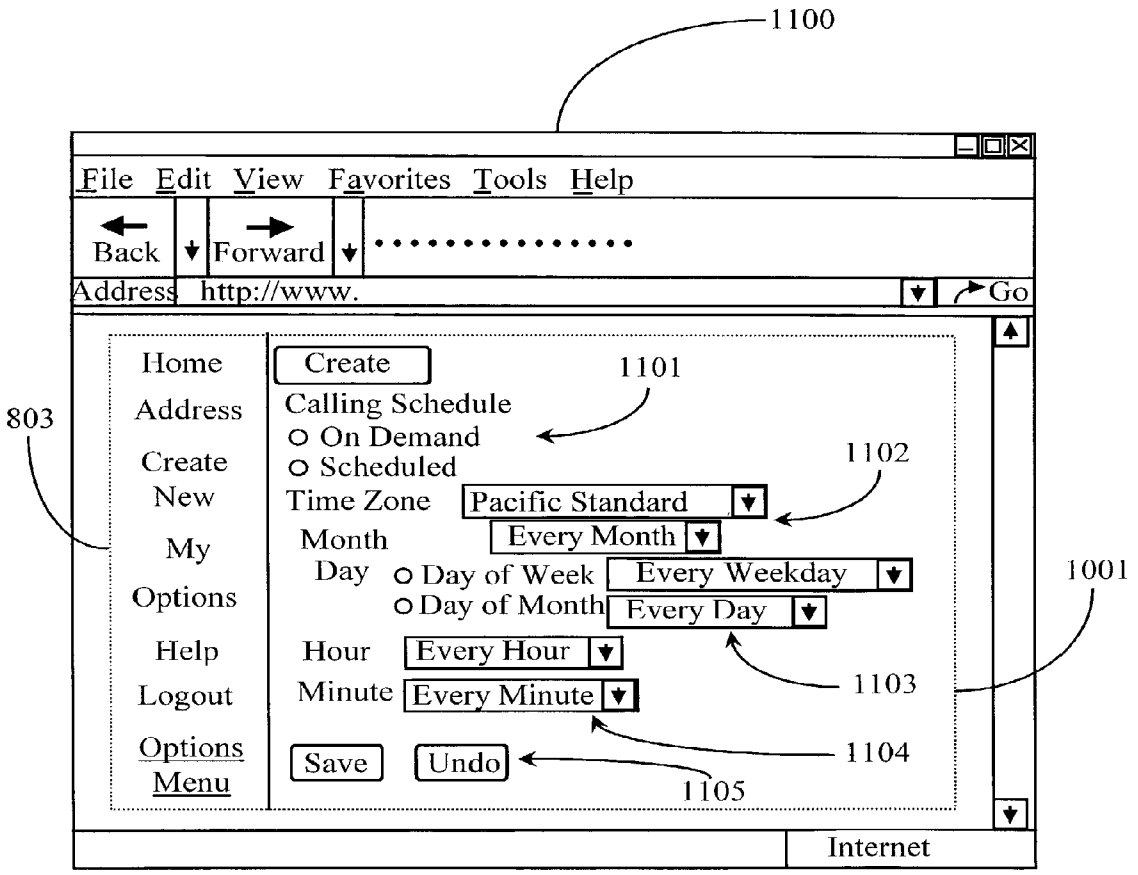
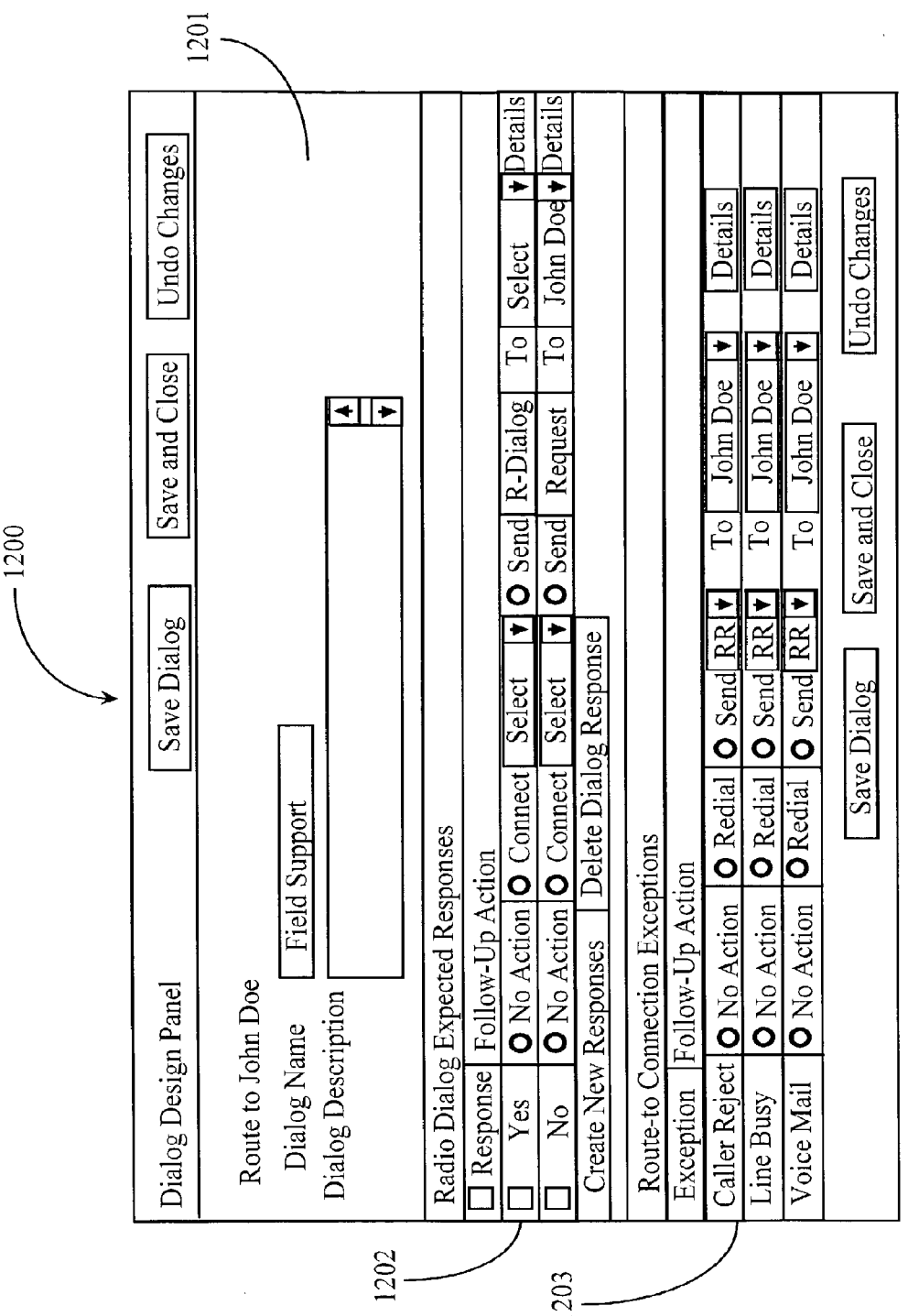
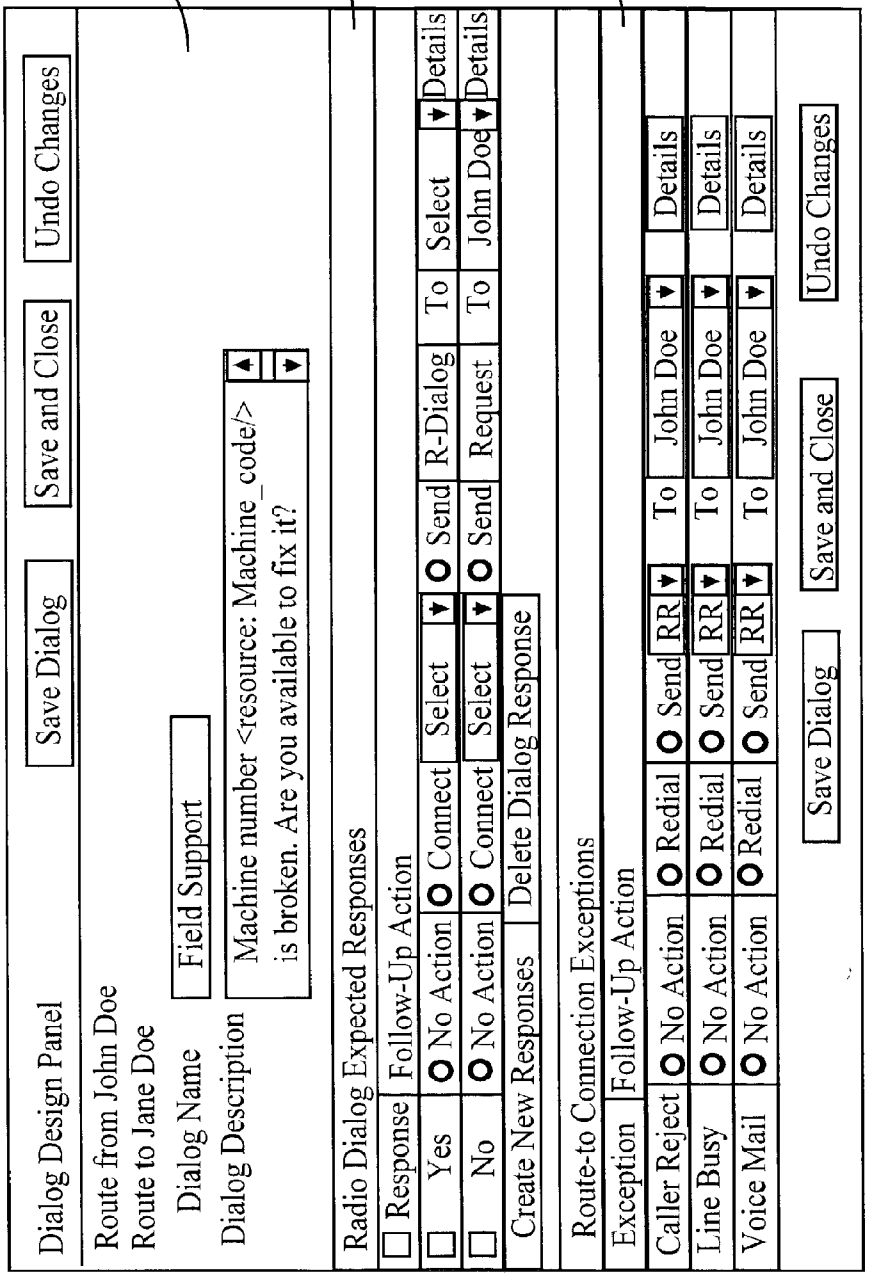

1100
File Edit View Favorites Tools Help
Back
Forward
Address http://www.
Go
Home
Address
Create New
My
Options
Help
Logout
Options Menu
803
Create
1101
Calling Schedule
On Demand
Scheduled
1102
Time Zone
Pacific Standard
Month
Every Month
Day
Day of Week
Every Weekday
Day of Month
Every Day
1001
Hour
Every Hour
1103
Minute
Every Minute
1104
Save
Undo
1105
Internet

Fig. 11



Dialog Design Window

Fig. 12



Dialog Design Window

Fig. 13

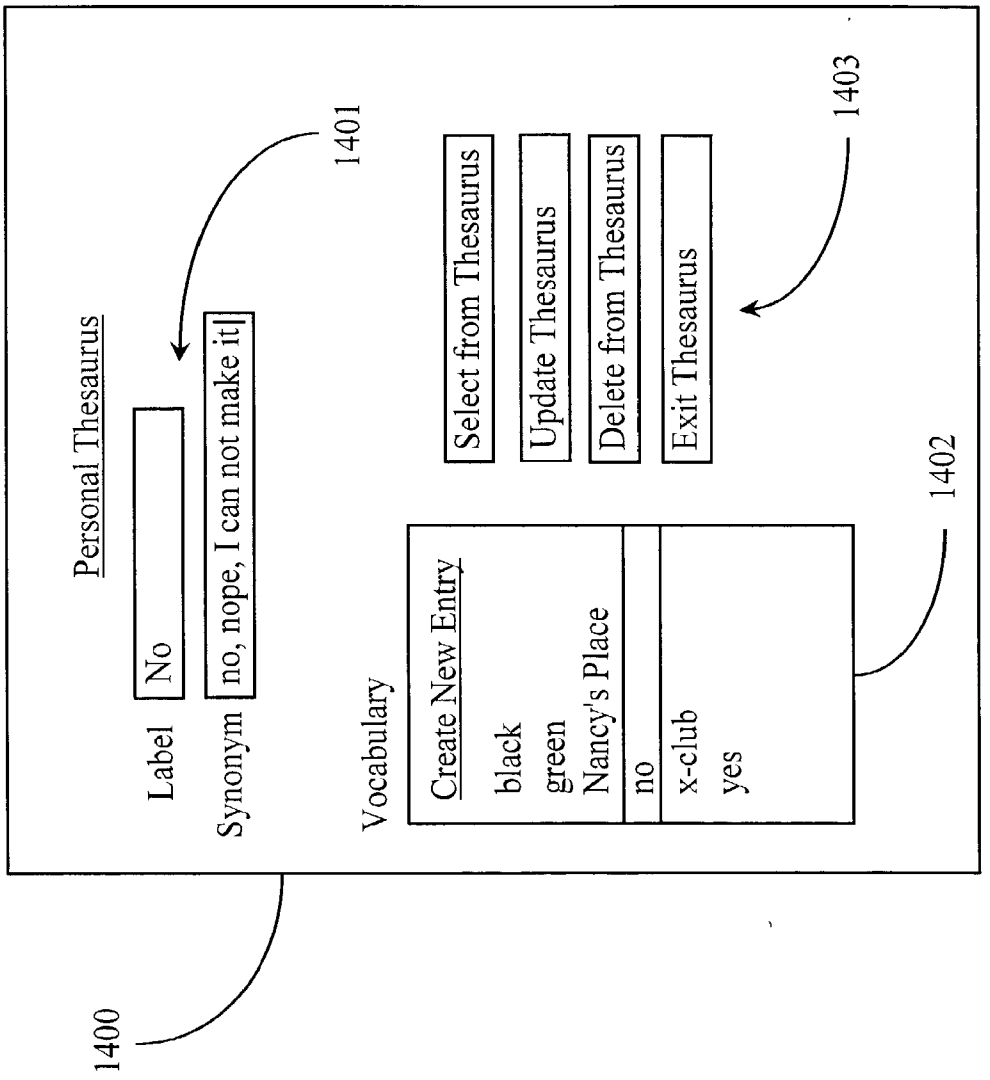

1400
Personal Thesaurus
Label
No
Synonym
no, nope, I can not make it
1401
Vocabulary
Create New Entry
black
green
Nancy's Place
no
x-club
yes
1402
Select from Thesaurus
Update Thesaurus
Delete from Thesaurus
Exit Thesaurus
1403

Fig. 14

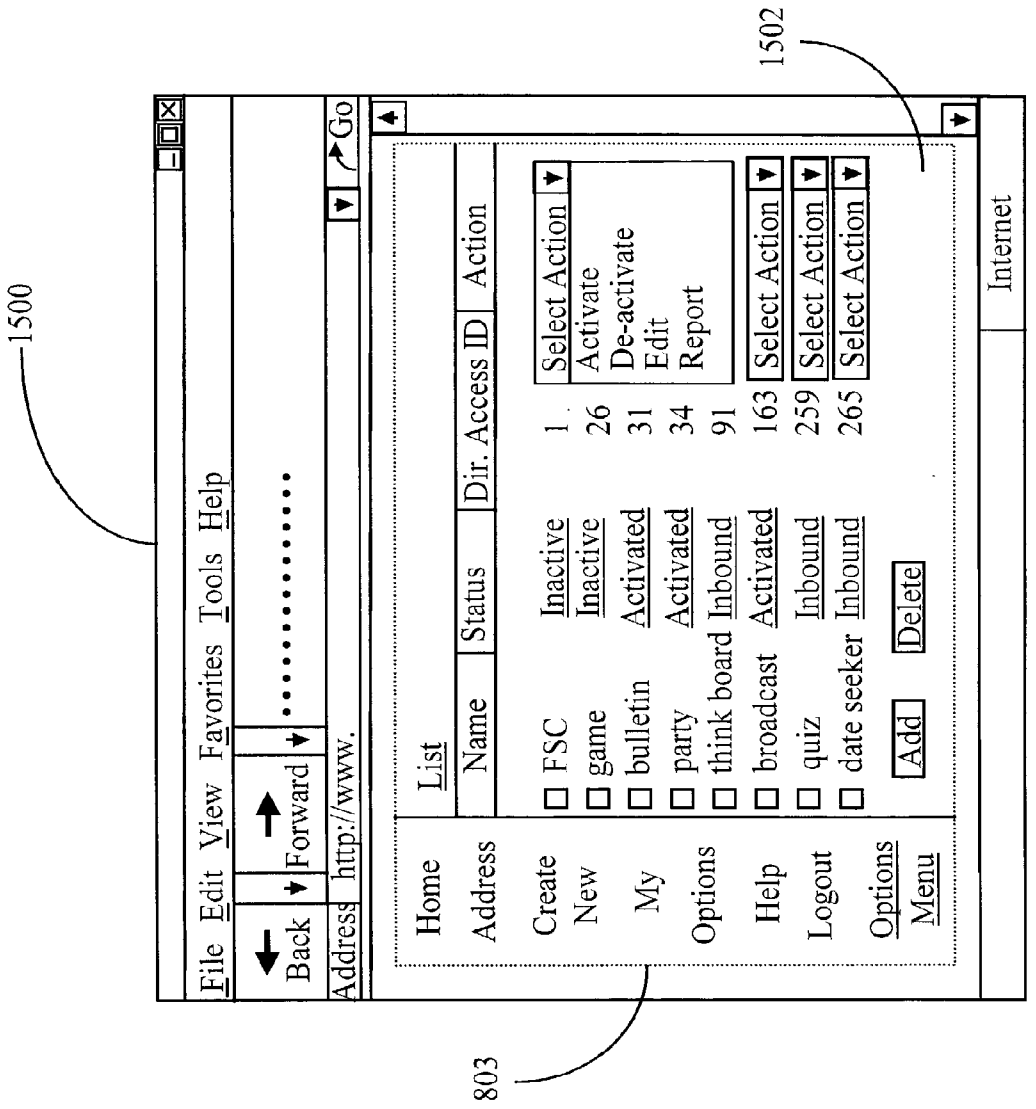

1500
File Edit View Favorites Tools Help
Back
Forward
Address http://www.
Go
Home
Address
List
Name
Status
Dir. Access ID
Action
Create New
My Options
Help
Logout
Options Menu
FSC Inactive 1
game Inactive 26
bulletin Activated 31
party Activated 34
think board Inbound 91
broadcast Activated 163
quiz Inbound 259
date seeker Inbound 265
Select Action
Activate
De-activate
Edit
Report
Add
Delete
Internet
803
1502

Fig. 15

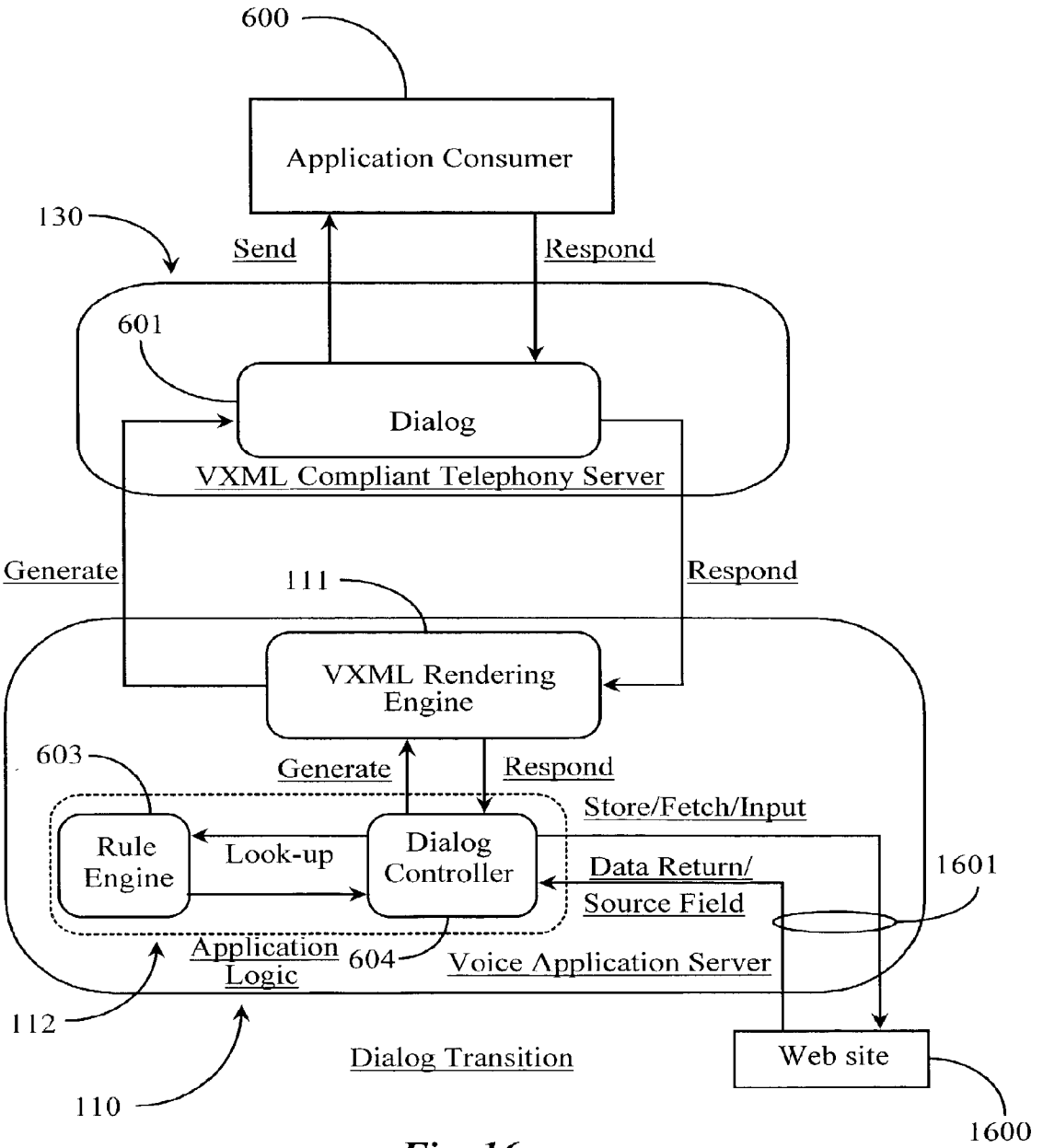

600
Application Consumer
130
Send
Respond
601
Dialog
VXML Compliant Telephony Server
Generate
Respond
111
VXML Rendering Engine
603
Generate
Respond
Store/Fetch/Input
Rule Engine
Look-up
Dialog Controller
Data Return/ Source Field
1601
Application Logic
604
Voice Application Server
112
110
Dialog Transition
Web site
1600

Fig. 16

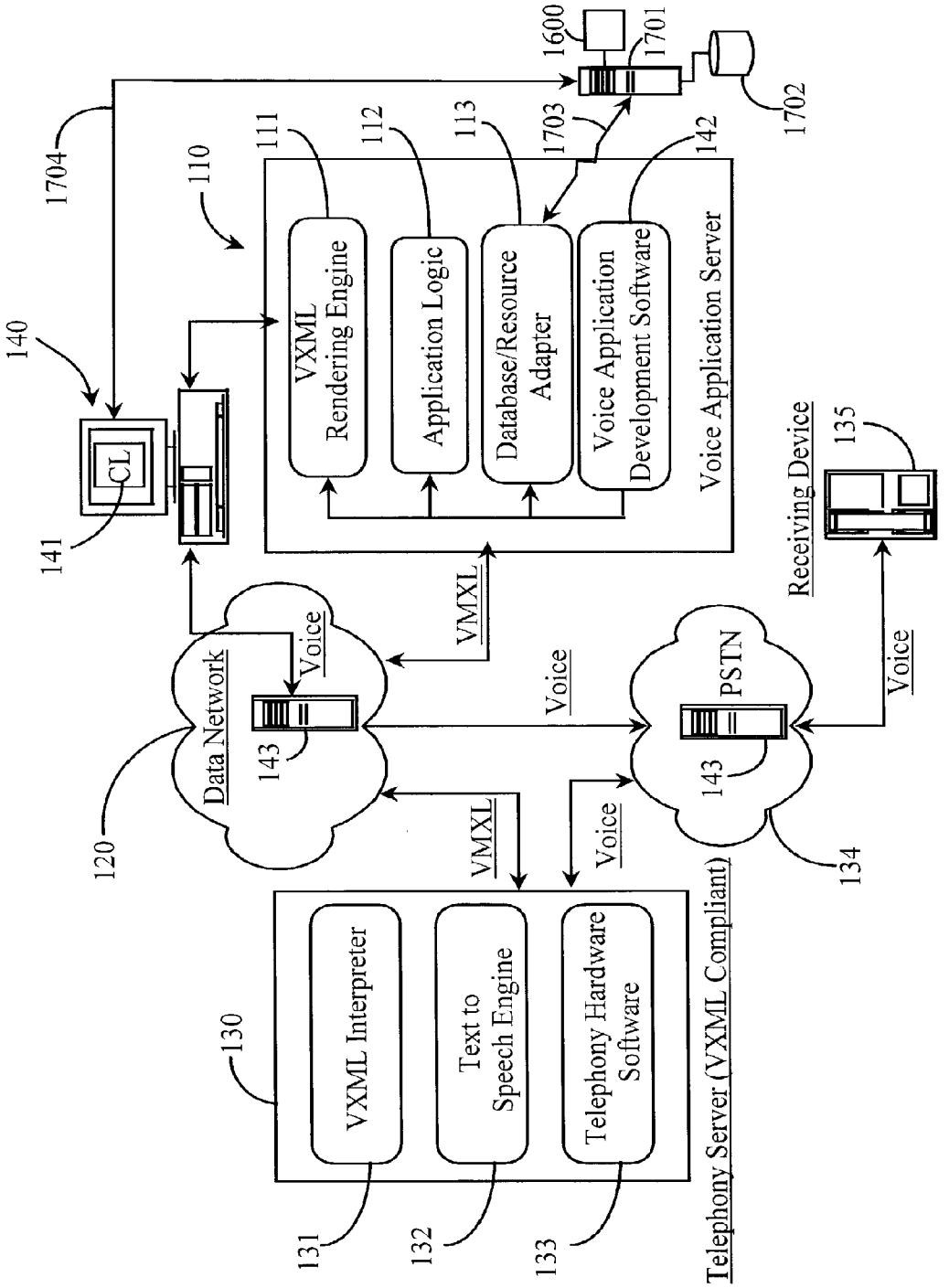

1600
1701
1702
1703
1704
110
111
112
113
142
VXML Rendering Engine
Application Logic
Database/Resource Adapter
Voice Application Development Software
Voice Application Server
140
CL
141
VMXL
Voice
Data Network
143
120
PSTN
143
134
Receiving Device
135
Voice
VMXL
Voice
130
VXML Interpreter
Text to Speech Engine
Telephony Hardware Software
131
132
133
Telephony Server (VXML Compliant)

Fig. 17

METHOD FOR AUTOMATED HARVESTING OF DATA FROM A WEB SITE USING A VOICE PORTAL SYSTEM

CROSS-REFERENCE TO RELATED DOCUMENTS

[0001] The present application is a Continuation of co-pending U.S. patent application Ser. No. 10/190,077, filed on Jul. 2, 2002, the disclosure of which is incorporated by reference herein. That application is a Continuation In Part of a U.S. patent application, Attorney docket No. P8100 entitled "Method and Apparatus for Development and Deployment of a Voice Software Application for Distribution to one or more Application Consumers", filed on Jun. 14, 2002, the disclosure of which is incorporated by reference herein in its entirety. That parent case filed on Jun. 14, 2002 claimed priority to provisional application Ser. No. 60/302,736 filed on Jul. 3, 2001, and incorporates all disclosure of that provisional application by reference. The present application therefore claims priority to both of the applications described above in this paragraph.

FIELD OF THE INVENTION

[0002] The present invention is in the area of software application development and pertains particularly to methods and apparatus for auto harvesting template-based web content using a VXML-enabled voice portal system.

BACKGROUND

[0003] A speech application is one of the most challenging applications to develop, deploy and maintain in a communications (typically telephony) environment. Expertise required for developing and deploying a viable application includes expertise in computer telephony integration (CTI) hardware and software, voice recognition software, text-to-speech software, and speech application logic.

[0004] With the relatively recent advent of voice extensive markup language (VXML) the expertise require to develop a speech solution has been reduced somewhat. VXML is a language that enables a software developer to focus on the application logic of the voice application without being required to configuring underlying telephony components. Typically, the developed voice application is run on a VXML interpreter that resides on and executes on the associated telephony system to deliver the solution.

[0005] As is shown in FIG. **1**A (prior art) a typical architecture of a VXML-compliant telephony system comprises a voice application server (**110**) and a VXML-compliant telephony server (**130**). Typical steps for development and deployment of a VXML enabled IVR solutions are briefly described below using the elements of FIG. **1**A.

[0006] Firstly, a new application database (**113**) is created or an existing one is modified to support VXML. Application logic **112** is designed in terms of workflow and adapted to handle the routing operations of the IVR system. VXML pages, which are results of functioning application logic, are rendered by a VXML rendering engine (**111**) based on a specified generation sequence.

[0007] Secondly, an object facade to server **130** is created comprising the corresponding VXML pages and is sent to server **130** over a network (**120**), which can be the Internet, an Intranet, or an Ethernet network. The VXML pages are integrated into rendering engine **111** such that they can be displayed according to set workflow at server **110**.

[0008] Thirdly, the VXML-telephony server **130** is configured to enable proper retrieval of specific VXML pages from rendering engine **111** within server **110**. A triggering mechanism is provided to server **110** so that when a triggering event occurs, an appropriate outbound call is placed from server **110**.

[0009] A VXML interpreter (**131**), a voice recognition text-to-speech engine (**132**), and the telephony hardware/software (**133**) are provided within server **130** and comprise server function. In prior art, the telephony hardware/software **130** along with the VXML interpreter **131** are packaged as an off-the-shelf IVR-enabling technology. Arguably the most important feature, however, of the entire system is the application server **110**. The application logic (**112**) is typically written in a programming language such as Java and packaged as an enterprise Java Bean archives The presentation logic required is handled by rendering engine **111** and is written in JSP or PERL.

[0010] An enhanced voice application system is known to the inventor and disclosed in the U.S. patent application entitled "Method and Apparatus for Development and Deployment of a Voice Software Application for Distribution to one or more Application Consumers" to which this application claims priority. That system uses a voice application server that is connected to a data network for storing and serving voice applications. The voice application server has a data connection to a network communications server connected to a communications network such as the well-known PSTN network. The communication server routes the created voice applications to their intended recipients.

[0011] A computer station is provided as part of the system and is connected to the data network and has access to the voice application server. A client software application is hosted on the computer station for the purpose of enabling users to create applications and manage their states. In this system, the user operates the client software hosted on the computer station in order to create voice applications through object modeling and linking. The applications, once created, are then stored in the application server for deployment. The user can control and manage deployment and state of deployed applications including scheduled deployment and repeat deployments in terms of intended recipients.

[0012] The system described above is quite advantageous in that a developer using the system need not be highly skilled in VXML language compilation, database programming, or advanced server design. Moreover, the system substantially reduces the amount of time allotted to project development. Still further, the system supports changes or modifications performed by anyone other than the developer without having to depend on the original author (because of his or her experience) to debug or make changes.

[0013] With the advent of the well known Internet network and growing technology to bridge two disparate networks such as the Internet and a PSTN network, the present inventors have concluded that Web data from Web sites should be a target data set for harvesting and conversion into a voice application that could then be routed to one or more users.

[0014] What is therefore clearly needed is a VMXL interface to a Web server that is capable of harvesting data from individual Web sites hosted on the server.

SUMMARY

[0015] In a preferred embodiment of the present invention a system for developing and deploying a voice application

using Web-based data as source data over a communications network to one or more recipients is provided, comprising a voice application server connected to a data network for storing and serving voice applications, the voice application server capable of accessing a network server and Website hosted therein, a network communications server connected to the data network and to the communications network for routing the voice applications to their intended recipients, a computer station connected to the data network having control access to at least the voice application server, the computer station capable of accessing the network server and Website hosted therein, and a software application running on the computer station for creating applications and managing their states. The system is characterized in that a developer operating the software application from the computer station accesses the target server and Website and creates at least one template for holding Web data according to logical order of site construction and links the templates to voice applications created by object modeling and linking, whereupon according to a triggering event, the triggered voice application accesses the intended Website and retrieves refreshed data from the site into the template and renders the retrieved data as voice delivered to intended recipients.

[0016] In a preferred embodiment the data network is the Internet network. Also in a preferred embodiment the voice application is delivered over a public-switched-telephony-network to a voice portal accessed by the intended recipients.

[0017] In some embodiments the security protocol used by the accessed Website is transferred as a voice access mechanism to the intended recipients. Also in some embodiments the data network connections of the computer station and of the voice application server are separate data network connections. In some cases the voice applications are scheduled for delivery at a specific time delivery made by outbound dialing and connection to a telephone of the intended recipient. The voice applications may be accessed by the intended recipients in real time.

[0018] In another aspect of the invention a voice application server for rendering voice dialog using Web-based data as source data is provided, comprising an instance of voice application development software, a database resource adapter, an instance of application logic;

[0019] a data rendering engine, and a network access line and software application for accessing servers and Websites hosted therein for the purpose of harvesting data there from for use in voice application creation and deployment. The server is characterized in that according to a trigger, the voice application server navigates to one or more servers and one or more Websites and obtains refreshed data there from converting the data to voice dialog and delivering the dialog to a voice portal accessible to an intended recipient.

[0020] In many preferred embodiments the network is the Internet network. Also in some preferred embodiments the voice application is delivered over a public-switched-telephony-network to a voice portal accessed by the intended recipients. In some cases the security protocol used by the accessed Website is transferred as a voice access mechanism to the intended recipients. Also in some cases the voice applications are scheduled for delivery at a specific time delivery made by outbound dialing and connection to a telephone of the intended recipient. The applications may be accessed by the intended recipients in real time.

[0021] In some embodiments of the invention a template is created for storing field and source data from a Website and the template is used in voice rendering in the server. The software for network access may be a browser application.

[0022] Embodiments of the invention taught in enabling detail below provide new and enhanced functionality for rendering information into audio form.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] FIG. **1**A is a block diagram illustrating a basic architecture of a VXML-enabled IVR development and deployment environment according to prior-art.

[0024] FIG. **1**B is a block diagram illustrating the basic architecture of FIG. **1**A enhanced to practice the present invention.

[0025] FIG. **2** is a process flow diagram illustrating steps for creating a voice application shell or container for a VXML voice application according to an embodiment of the present invention.

[0026] FIG. **3** is a block diagram illustrating a simple voice application container according to an embodiment of the present invention.

[0027] FIG. **4** is a block diagram illustrating a dialog object model according to an embodiment of the present invention.

[0028] FIG. **5** is a process flow diagram illustrating steps for voice dialog creation for a VXML-enabled voice application according to an embodiment of the present invention.

[0029] FIG. **6** is a block diagram illustrating a dialog transition flow after initial connection with a consumer according to an embodiment of the present invention.

[0030] FIG. **7** is a plan view of a developer's frame containing a developer's login screen of according to an embodiment of the present invention.

[0031] FIG. **8** is a plan view of a developer's frame containing a screen shot of a home page of the developer's platform interface of FIG. **7**.

[0032] FIG. **9** is a plan view of a developer's frame containing a screen shot of an address book **911** accessible through interaction with the option Address in section **803** of the previous frame of FIG. **8**.

[0033] FIG. **10** is a plan view of a developer's frame displaying a screen **1001** for creating a new voice application.

[0034] FIG. **11** is a plan view of a developer's frame illustrating screen of FIG. **10** showing further options as a result of scrolling down.

[0035] FIG. **12** is a screen shot of a dialog configuration window illustrating a dialog configuration page according to an embodiment of the invention.

[0036] FIG. **13** is a screen shot **1300** of dialog design panel of FIG. **12** illustrating progression of dialog state to a subsequent contact.

[0037] FIG. **14** is a screen shot of a thesaurus configuration window activated from the example of FIG. **13** according to a preferred embodiment.

[0038] FIG. **15** is a plan view of a developer's frame illustrating a screen for managing created modules according to an embodiment of the present invention.

[0039] FIG. **16** is a block diagram of the dialog transition flow of FIG. **6** enhanced for Web harvesting according to an embodiment of the present invention.

[0040] FIG. **17** is a block diagram of the voice application distribution environment of FIG. **1**B illustrating added com-

ponents for automated Web harvesting and data rendering according to an embodiment of the present invention.

DETAILED DESCRIPTION

[0041] According to preferred embodiments of the present invention, the inventor teaches herein, in an enabling fashion, a novel system for developing and deploying real-time dynamic or static voice applications in an object-oriented way that enables inbound or outbound delivery of IVR and other interactive voice solutions in supported communications environments.

[0042] FIG. **1**A is a block diagram illustrating a basic architecture of a VXML-enabled IVR development and deployment environment according to prior art. As described with reference to the background section, the prior-art architecture of this example is known to and available to the inventor. Developing and deploying voice applications for the illustrated environment, which in this case is a telephony environment, requires a very high level of skill in the art. Elements of this prior-art example that have already been introduced with respect to the background section of this specification shall not be re-introduced.

[0043] In this simplified scenario, voice application server **110** utilizes database/resource adapter **113** for accessing a database or other resources for content. Application logic **112** comprising VXML script, business rules, and underlying telephony logic must be carefully developed and tested before single applications can be rendered by rendering engine **111**. Once voice applications are complete and servable from server **110**, they can be deployed through data network **120** to telephony server **130** where interpreter **131** and text-to speech engine **132** are utilized to formulate and deliver the voice application in useable or playable format for telephony software and hardware **133**. The applications are accessible to a receiving device, illustrated herein as device **135**, a telephone, through the prevailing network **134**, which is in this case a public-switched-telephone-network (PSTN) linking the telephony server to the consumer (device **135**) generally through a telephony switch (not shown).

[0044] Improvements to this prior-art example in embodiments of the present invention concern and are focused in the capabilities of application server **110** with respect to development and deployment issues and with respect to overall enhancement to response capabilities and options in interaction dialog that is bi-directional. Using the description of existing architecture deemed state-of-art architecture, the inventor herein describes additional components that are not shown in the prior-art example of FIG. **1**A, but are illustrated in a novel version of the example represented herein by FIG. **1**B.

[0045] FIG. **1**B is a block diagram illustrating the basic architecture of FIG. **1**A enhanced to illustrate an embodiment of the present invention. Elements of the prior-art example of FIG. **1**A that are also illustrated in FIG. **1**B retain their original element numbers and are not re-introduced. For reference purposes an entity (a person) that develops a voice application shall be referred to hereinafter in this specification as either a producer or developer.

[0046] A developer or producer of a voice application according to an embodiment of the present invention operates preferably from a remote computerized workstation illustrated herein as station **140**. Station **140** is essentially a network-connected computer station. Station **140** may be housed within the physical domain also housing application server **110**. In another embodiment, station **140** and application server **110** may reside in the same machine. In yet another embodiment, a developer may operate station **140** from his or her home office or from any network-accessible location including any wireless location.

[0047] Station **140** is equipped with a client software tool (CL) **141**, which is adapted to enable the developer to create and deploy voice applications across the prevailing system represented by servers **110**, **130**, and by receiving device **135**. CL **141** is a Web interface application similar to or incorporated with a Web browser application in this example, however other network situations may apply instead. CL **141** contains the software tools required for the developer to enable enhancements according to embodiments of the invention. Station **140** is connected to a voice portal **143** that is maintained either on the data network (Internet, Ethernet, Intranet, etc.) and/or within telephony network **134**. In this example portal **143** is illustrated logically in both networks. Voice portal **143** is adapted to enable a developer or a voice application consumer to call in and perform functional operations (such as access, monitor, modify) on selected voice applications.

[0048] Within application server **110** there is an instance of voice application development server **142** adapted in conjunction with the existing components **111**-**113** to provide dynamic voice application development and deployment according to embodiments of the invention.

[0049] Portal **143** is accessible via network connection to station **140** and via a network bridge to a voice application consumer through telephony network **134**. In one example, portal **143** is maintained as part of application server **110**. Portal **143** is, in addition to an access point for consumers is chiefly adapted as a developer's interface server. Portal **143** is enabled by a SW instance **144** adapted as a server instance to CL **141**. In a telephony embodiment, portal **143** may be an interactive voice response (IVR) unit.

[0050] In a preferred embodiment, the producer or developer of a voice application accesses application server **110** through portal **143** and data network **120** using remote station **140** as a "Web interface" and first creates a list of contacts. In an alternative embodiment, station **140** has direct access to application server **110** through a network interface. Contacts are analogous to consumers of created voice applications. CL **141** displays, upon request and in order of need, all of the required interactive interfaces for designing, modifying, instantiating, and executing completed voice applications to launch from application server **110** and to be delivered by server **130**.

[0051] The software of the present invention enables voice applications to be modeled as a set of dialog objects having business and telephony (or other communication delivery/access system) rules as parameters without requiring the developer to perform complicated coding operations. A dialog template is provided for modeling dialog states. The dialog template creates the actual speech dialog, specifies the voice application consumer (recipient) of the dialog, captures the response from the voice application consumer and performs any follow-up actions based upon system interpretation of the consumer response. A dialog is a reusable component and can be linked to a new dialog or to an existing (stored) dialog. A voice application is a set of dialogs inter-linked by a set of business rules defined by the voice application producer. Once the voice application is completed, it is deployed

by server **110** and is eventually accessible to the authorized party (device **135**) through telephony server **130**.

[0052] The voice applications are in a preferred embodiment in the form of VXML to run on VXML-compliant telephony server **130**. This process is enabled through VXML rendering engine **111**. Engine **111** interacts directly with server **130**, locates the voice application at issue, retrieves its voice application logic, and dynamically creates the presentation in VXML and forwards it to server **130** for processing and delivery. Once interpreter **131** interprets the VXML presentation it is sent to or accessible to device **135** in the form of an interactive dialog (in this case an IVR dialog). Any response from device **135** follows the same path back to application server **110** for interpretation by engine **111**. Server **110** then retrieves the voice application profile from the database accessible through adapter **113** and determines the next business rule to execute locally. Based upon the determination a corresponding operation associated with the rule is taken. A next (if required) VXML presentation is then forwarded to rendering engine **111**, which in turn dynamically generates the next VXML page for interpretation, processing and deployment at server **130**. This two-way interaction between the VXML-compliant telephony server (**130**) and the voice application server (**110**) continues in the form of an automated logical sequence of VXML dialogs until the voice application finally reaches its termination state.

[0053] A voice application (set of one or more dialogs) can be delivered to the consumer (target audience) in outbound or inbound fashion. For an inbound voice application, a voice application consumer calls in to voice portal **143** to access the inbound voice application served from server **130**. The voice portal can be mapped to a phone number directly or as an extension to a central phone number. In a preferred embodiment the voice portal also serves as a community forum where voice application producers can put their voice applications into groups for easy access and perform operational activities such as voice application linking, reporting, and text-to-speech recording and so on.

[0054] For an outbound voice application there are two sub-types. These are on-demand outbound applications and scheduled outbound applications. For on-demand outbound applications server **110** generates an outbound call as soon as the voice application producer issues an outbound command associated with the application. The outbound call is made to the target audience and upon the receipt of the call the voice application is launched from server **130**. For scheduled outbound applications, the schedule server (not shown within server **110**) launches the voice application as soon as the producer-specified date and time has arrived. In a preferred embodiment both on-demand and scheduled outbound application deployment functions support unicast, multicast and broadcast delivery schemes.

[0055] As described above, a voice application created by application server **110** consists of one or more dialogs. The contents of each dialog can be static or dynamic. Static content is content sourcing from the voice application producer. The producer creates the contents when the voice application is created. Dynamic content sources from a third-party data source.

[0056] In a preferred embodiment a developers tool contains an interactive dialog design panel (described in detail later) wherein a producer inputs a reference link in the form of eXtensible Markup Language (XML) to the dialog description or response field. When a dialog response is executed and interpreted by application server **110**, the reference link invokes a resource Application-Program-Interface (API) that is registered in resource adapter **113**. The API goes out in real time and retrieves the requested data and integrates the returned data into the existing dialog. The resulting and subsequent VXML page being generated has the dynamic data embedded onto it.

[0057] One object of the present invention is a highly dynamic, real time IVR system that tailors itself automatically to the application developer's specified data source requirement. Another object of the present invention is to enable rapid development and deployment of a voice application without requirement of any prior knowledge of VXML or any other programming technologies. A further object of the present invention is to reduce the typical voice application production cycle and drastically reduce the cost of production.

[0058] FIG. **2** is a process flow diagram illustrating steps for creating a voice application shell or container for a VXML voice application according to an embodiment of the present invention. A developer utilizing a client application known as a thin client analogous to CL **141** on station **140** described with reference to FIG. **1***b*, creates a voice application shell or voice application container. At step **201** the developer logs in to the system at a login page. At step **202** the developer creates a contact list of application consumers. Typically a greeting or welcome page would be displayed before step **202**. An application consumer is an audience of one or more entities that would have access to and interact with a voice application. A contact list is first created so that all of the intended contacts are available during voice application creation if call routing logic is required later on. The contact list can either be entered individually in the event of more than one contact by the producer or may be imported as a set list from some organizer/planner software, such as Microsoft Outlook™ or perhaps a PDA™ organizer.

[0059] In one embodiment of the present invention the contact list may reside on an external device accessed by a provided connector (not shown) that is configured properly and adapted for the purpose of accessing and retrieving the list. This approach may be used, for example, if a large, existing customer database is used. Rather than create a copy, the needed data is extracted from the original and provided to the application.

[0060] At step **203**, a voice application header is populated. A voice application header is simply a title field for the application. The field contains a name for the application and a description of the application. At step **204**, the developer assigns either and inbound or outbound state for the voice application. An outbound application is delivered through an outbound call while the consumer accesses an inbound voice application.

[0061] In the case of the inbound application, in step **205** the system sets a default addressee for inbound communications. The developer selects a dialog from a configured list in step **206**. It is assumed in this example that the dialogs have already been created. At step **207**, the developer executes the dialog and it is deployed automatically.

[0062] In the case of an outbound designation in step **204**, the developer chooses a launch type in step **208**. A launch type can be either an on-demand type or a scheduled type. If the choice made by the developer in step **208** is scheduled, then in step **209**, the developer enters all of the appropriate time and date parameters for the launch including parameters for recur-

ring launches of the same application. In the case of an on demand selection for application launch in step **208**, then in step **210** the developer selects one or more contacts from the contact list established in step **202**. It is noted herein that step **210** is also undertaken by the developer after step **209** in the case of a scheduled launch. At step **207**, the dialog is created. In this step a list of probable dialog responses for a voice application wherein interaction is intended may also be created and stored for use.

[0063] In general sequence, a developer creates a voice application and integrates the application with a backend data source or, optionally, any third party resources and deploys the voice application. The application consumer then consumes the voice application and optionally, the system analyzes any consumer feedback collected by the voice application for further interaction if appropriate. The steps of this example pertain to generating and launching a voice application from "building blocks" that are already in place.

[0064] FIG. **3** is a block diagram illustrating a simple voice application container **300** according to an embodiment of the present invention. Application container **300** is a logical container or "voice application object" **300**. Also termed a shell, container **300** is logically illustrated as a possible result of the process of FIG. **2** above. Container **300** contains one or more dialog states illustrated herein as dialogs **301** a-n labeled in this example as dialogs **1-4**. Dialogs **301***a-n* are objects and therefore container **300** is a logical grouping of the set of dialog objects **301***a-n*.

[0065] The represented set of dialog objects **301***a-n* is interlinked by business rules labeled rules **1-4** in this example. Rules **1-4** are defined by the developer and are rule objects. It is noted herein that that there may be many more or fewer dialog objects **301***a-n* as well as interlinking business rule objects **1-4** comprising container object **300** without departing from the spirit and scope of the present invention. The inventor illustrates 4 of each entity and deems the representation sufficient for the purpose of explaining the present invention.

[0066] In addition to the represented objects, voice application shell **300** includes a plurality of settings options. In this example, basic settings options are tabled for reference and given the element number **305** a-c illustrating 3 listed settings options. Reading in the table from top to bottom, a first setting launch type (**305***a*) defines an initial entry point for voice application **300** into the communications system. As described above with reference to FIG. **2** step **204**, the choices for launch type **305***a* are inbound or outbound. In an alternative embodiment, a launch type may be defined by a third party and be defined in some other pattern than inbound or outbound.

[0067] Outbound launch designation binds a voice application to one or more addressees (consumers). The addressee may be a single contact or a group of contacts represented by the contact list or distribution list also described with reference to FIG. **2** above (step **202**). When the outbound voice application is launched in this case, it is delivered to the addressee designated on a voice application outbound contact field (not shown). All addressees designated receive a copy of the outbound voice application and have equal opportunity to interact (if allowed) with the voice application dialog and the corresponding backend data resources if they are used in the particular application.

[0068] In the case of an inbound voice application designation for launch type **305***a*, the system instructs the application to assume a ready stand-by mode. The application is launched when the designated voice application consumer actively makes a request to access the voice application. A typical call center IVR system assumes this type of inbound application.

[0069] Launch time setting (**305***b*) is only enabled as an option if the voice application launch type setting **305***a* is set to outbound. The launch time setting is set to instruct a novel scheduling engine, which may be assumed to be part of the application server function described with reference to FIG. **1**B. The scheduling engine controls the parameter of when to deliver of when to deliver the voice application to the designated addressees. The time setting may reflect on-demand, scheduled launch, or any third-party-defined patterns.

[0070] On-demand gives the developer full control over the launch time of the voice application. The on-demand feature also allows any third-party system to issue a trigger event to launch the voice application. It is noted herein that in the case of third-party control the voice application interaction may transcend more than one communications system and or network.

[0071] Property setting **305***c* defines essentially how the voice application should behave in general. Possible state options for setting **305***c* are public, persistent, or sharable. A public state setting indicates that the voice application should be accessible to anyone within the voice portal domain so that all consumers with minimum privilege can access the application. A persistent state setting for property **305***c* ensures that only one copy of the voice application is ever active regardless of how many consumers are attempting to access the application. An example of such a scenario would be that of a task-allocation voice application. For example, in a task-allocation scenario there are only a number of time slots available for a user to access the application. If the task is a request from a pool of contacts such as perhaps customer-support technicians to lead a scheduled chat session, then whenever a time slot has been selected, the other technicians can only select the slots that are remaining. Therefore if there is only one copy of the voice application circulating within the pool of technicians, the application captures the technician's response on a first-come first-serve basis.

[0072] A sharable application state setting for property **305***a* enables the consumer to "see" the responses of other technicians in the dialog at issue, regardless of whether the voice application is persistent or not. Once the voice application shell is created, the producer can then create the first dialog of the voice application as described with reference to FIG. **2** step **207**. It is reminded herein that shell **300** is modeled using a remote and preferably a desktop client that will be described in more detail later in this specification.

[0073] FIG. **4** is a block diagram illustrating a dialog object model **400** according to an embodiment of the present invention. Dialog object model **400** is analogous to any of dialog objects **301***a-n* described with reference to FIG. **3** above. Object **400** models a dialog and all of its properties. A properties object illustrated within dialog object **400** and labeled Object Properties (**410**) contains the dialog type and properties including behavior states and business rules that apply to the dialog.

[0074] For example, every dialog has a route-to property illustrated in the example as Route To property (**411**). Property **411** maps to and identifies the source of the dialog. Similarly, every dialog has a route-from property illustrated

herein as Route From property (**412**). Route from property **412** maps to and identifies the recipient contact of the dialog or the dialog consumer.

[0075] Every dialog falls under a dialog type illustrated in this example by a property labeled Dialog Type and given the element number **413**. Dialog type **413** may include but is not limited to the following types of dialogs:

[0076] 1. Radio Dialog: A radio dialog allows a voice application consumer to interactively select one of available options from an option list after hearing the dialog description.

[0077] 2. Bulletin Dialog: A bulletin dialog allows a voice application consumer to interact with a bulletin board-like forum where multiple consumers can share voice messages in an asynchronous manner.

[0078] 3. Statement Dialog: A statement dialog plays out a statement to a voice application consumer without expecting any responses from the consumer.

[0079] 4. Open Entry Dialog: An open entry dialog allows a voice application consumer to record a message of a pre-defined length after hearing the dialog description.

[0080] 5. Third Party Dialog: A third party dialog is a modular container structure that allows the developer to create a custom-made dialog type with its own properties and behaviors. An example would be Nuance's SpeechObject™.

[0081] Each dialog type has one or more associated business rules tagged to it enabling determination of a next step in response to a perceived state. A rule compares the application consumer response with an operand defined by the application developer using an operational code such as less than, greater than, equal to, or not equal to. In a preferred embodiment of the invention the parameters surrounding a rule are as follows:

[0082] If user response is equal to the predefined value, then perform one of the following:

[0083] A. Do nothing and terminate the dialog state.

[0084] B. Do a live bridge transfer to the contact specified. Or,

[0085] C. Send another dialog to another contact.

[0086] In the case of an outbound voice application, there are likely to be exception-handling business rules associated with perceived states. In a preferred embodiment of the present invention, exception handling rules are encapsulated into three different events:

[0087] 1. An application consumer designated to receive the voice application rejects a request for interacting with the voice application.

[0088] 2. An application consumer has a busy connection at the time of launch of the voice application, for example, a telephone busy signal. And,

[0089] 3. An application consumer's connection is answered by or is redirected to a non-human device, for example, a telephone answering machine.

[0090] For each of the events above, any one of the three follow-up actions are possible according to perceived state:

[0091] 1. Do nothing and terminate the dialog state.

[0092] 2. Redial the number.

[0093] 3. Send another dialog to another contact.

[0094] FIG. **5** is a process flow diagram illustrating steps for voice dialog creation for a VXML-enabled voice application according to an embodiment of the present invention. All dialogs can be reused for subsequent dialog routing. There is, as previously described, a set of business rules for every dialog and contact pair. A dialog be active and be able to transit from one dialog state to another only when it is rule enabled.

[0095] At step **501***a* developer populates a dialog description field with a dialog description. A dialog description may also contain reference to XML tags as will be described further below. At step **502**, parameters of the dialog type are entered based on the assigned type of dialog. Examples of the available parameters were described with reference to FIG. **4** above.

[0096] At step **503** the developer configures the applicable business rules for the dialog type covering, as well, follow up routines. In one embodiment rules configuration at step **503** resolves to step **505** for determining follow-up routines based on the applied rules. For example, the developer may select at step **505**, one of three types of transfers. For example, the developer may configure for a live transfer as illustrated by step **506**; transfer to a next dialog for creation as illustrated by step **507**; or the developer may configure for dialog completion as illustrated by step **508**.

[0097] If the developer does not branch off into configuring sub-routines **506**, **507**, or **508** from step **505**, but rather continues from step **503** to step **504** wherein inbound or outbound designation for the dialog is system assigned, then the process must branch from step **504** to either step **508** or **509**, depending on whether the dialog is inbound or outbound. If at step **504**, the dialog is inbound, then at step **508** the dialog is completed. If the assignment at step **504** is outbound, then at step **509** to configure call exception business rules.

[0098] At step **510**, the developer configures at least one follow-up action for system handling of exceptions. If no follow-up actions are required to be specified at step **510**, then the process resolves to step **508** for dialog completion. If an action or actions are configured at step **510**, then at step **511** the action or actions are executed such as a system re-dial, which the illustrated action for step **511**.

[0099] In a preferred embodiment, once the voice application has been created, it can be deployed and accessed through the telephone. The method of access, of course, depends on the assignment configured at step **504**. For example, if the application is inbound, the application consumer accesses a voice portal to access the application. As described further above, a voice portal is a voice interface for accessing a selected number of functions of the voice application server described with reference to FIG. **1**B above. A voice portal may be a connection-oriented-switched-telephony (COST) enabled portal or a data-network-telephony (DNT) enabled portal. In the case of an outbound designation at step **504**, the application consumer receives the voice application through an incoming call to the consumer originated from the voice application server. In a preferred embodiment, the outbound call can be either COST based or DNT based depending on the communications environment supported.

[0100] FIG. **6** is a block diagram illustrating a dialog transition flow after initial connection with a consumer according to an embodiment of the present invention. Some of the elements illustrated in this example were previously introduced with respect to the example of FIG. **1**B above and therefore shall retain their original element numbers. In this example, an application consumer is logically illustrated as Application Consumer **600** that is actively engaged in interaction with a dialog **601** hosted by telephony server **130**.

Server **130** is, as previously described a VMXL compliant telephony server as is so labeled.

[0101] Application server **110** is also actively engaged in the interaction sequence and has the capability to provide dynamic content to consumer **600**. As application consumer **600** begins to interact with the voice application represented herein by dialog **600** within telephony server **130**, voice application server **110** monitors the situation. In actual practice, each dialog processed and sent to server **130** for delivery to or access by consumer **600** is an atomic unit of the particular voice application being deployed and executed. Therefore dialog **601** may logically represent more than one single dialog.

[0102] In this example, assuming more than one dialog, dialog **601** is responsible during interaction for acquiring a response from consumer **600**. Arrows labeled Send and Respond represent the described interaction. When consumer **600** responds to dialog content, the response is sent back along the same original path to VXML rendering engine **111**, which interprets the response and forwards the interpreted version to a provided dialog controller **604**. Controller **604** is part of application logic **112** in server **110** described with reference to FIG. **1**B. Dialog controller **604** is a module that has the ability to perform table lookups, data retrieve and data write functions based on established rules and configured response parameters.

[0103] When dialog controller **604** receives a dialog response, it stores the response corresponding to the dialog at issue (**601**) to a provided data source **602** for data mining operations and workflow monitoring. Controller **604** then issues a request to a provided rules engine **603** to look-up the business rule or rules that correspond to the stored response. Once the correct business rule has been located for the response, the dialog controller starts interpretation. If the business rule accessed requires reference to a third-party data source (not shown), controller **604** makes the necessary data fetch from the source. Any data returned by controller **604** is integrated into the dialog context and passed onward VXML rendering engine **111** for dialog page generation of a next dialog **601**. The process repeats until dialog **601** is terminates.

[0104] In one embodiment, the business rule accessed by controller **604** as a result of a received response from consumer **600** carries a dialog transition state other than back to the current application consumer. In this case controller **604** spawns an outbound call from application server **110** to deliver the next or "generated dialog" to the designated target application consumer. At the same time, the current consumer has his/her dialog state completed as described with reference to FIG. **5** step **508** according to predefined logic specified in the business rule.

[0105] It will be apparent to one with skill in the art that a dialog can contain dynamic content by enabling controller **604** to have access to data source **602** according to rules served by rule engine **603**. In most embodiments there are generally two types of dynamic content. Both types are, in preferred embodiments, structured in the form of XML and are embedded directly into the next generated dialog page. The first of the 2 types of dynamic content is classified as non-recurring. Non-recurring content makes a relative reference to a non-recurring resource label in a resource adapter registry within a resource adapter analogous to adapter **113** of voice application server **110** described with reference to FIG. **1**B.

[0106] In the above case, when dialog controller **604** interprets the dialog, it first scans for any resource label. If a match is found, it looks up the resource adapter registry and invokes the corresponding resource API to fetch the required data into the new dialog context. Once the raw data is returned from the third-party data source, it passes the raw data to a corresponding resource filter for further processing. When completed in terms of processing by the filter, the dialog resource label or tag is replaced with the filtered data and is integrated transparently into the new dialog.

[0107] The second type of dynamic content is recurring. Recurring content usually returns more than one set of a name and value pair. An example would be a list of stocks in an application consumer's stock portfolio. For example, a dialog that enables consumer **600** to parrot a specific stock and have the subsequent quote returned through another dialog state is made to use recurring dynamic content to achieve the desired result. Recurring content makes a relative reference to a recurring resource label in the resource adapter registry of voice application server **110**. When controller **604** interprets the dialog, it handles the resource in an identical manner to handling of non-recurring content. However, instead of simply returning the filtered data back to the dialog context, it loops through the data list and configures each listed item as a grammar-enabled keyword. In so doing, consumer **600** can parrot one of the items (separate stocks) in the list played in the first dialog and have the response captured and processed for return in the next dialog state. The stock-quote example presented below illustrates possible dialog/response interactions from the viewpoint of consumer **600**.

[0108] Voice Application: "Good morning Leo, what stock quote do you want?"

[0109] Application Consumer: "Oracle"

[0110] Voice Application: "Oracle is at seventeen dollars."

[0111] Voice Application: "Good morning Leo, what stock quote do you want?"

[0112] This particular example consists of two dialogs.

[0113] The first dialog plays out the statement "Good morning Leo, what stock quote do you want?" The dialog is followed by a waiting state that listens for keywords such as Oracle, Sun, Microsoft, etc. The statement consists of two dynamic non-recurring resource labels. The first one is the time in day: Good morning, good afternoon, or good evening. The second dynamic content is the name of the application consumer. In this case, the name of the consumer is internal to the voice application server, thus the type of the resource label is SYSTEM. In the actual dialog description field, it may look something like this:

[0114] <resource type='ADAPTER' name='time greeting'/> <resource type='SYSTEM' name='target_contact'/>, what stock quote do you want?

[0115] Because the dialog is expecting the consumer to say a stock out of his/her existing portfolio, the dialog type is radio dialog, and the expected response property of the radio dialog is

```
<resource type='ADAPTER' name='stock_list'>
<param>
<resource type='SYSTEM' name='target_contact_id'/>
    </param>
</resource>
```

[0116] This XML resource label tells dialog controller **604** to look for a resource label named stock_list and to invoke the corresponding API with target_contact_id as the parameter. Upon completion of the data fetching, the list of stocks is integrated into the dialog as part of the grammars. And whatever the user responds to in terms of stock identification is matched against the grammars at issue (stocks in portfolio) and assigned the grammar return value to the dialog response, which can then forward it to the next dialog as resource of DIALOG type.

[0117] The producer can make reference to any dialog return values in any subsequent dialog by using <resource type='DIALOG' name='dialog_name'/>. This rule enables the producer to play out the options the application consumer selected previously in any follow-up dialogs.

[0118] The second dialog illustrated above plays out the quote of the stock selected from the first dialog, then returns the flow back to the first dialog. Because no extra branching logic is involved in this dialog, the dialog type in this case is a statement dialog. The dialog's follow-up action is simply to forward the flow back to the first dialog. In such a case, the dialog statement is:

```
<resource type='DIALOG' name='select stock dialog'/>
<resource type='ADAPTER' name='get_stock_quote'>
<param>
<resource type='DIALOG' name='select stock dialog'/>
    </param>
</resource>
```

[0119] Besides making reference to ADAPTER, DIALOG and SYSTEM type, the dialog can also take in other resource types such as SOUND and SCRIPT. SOUND can be used to impersonate the dialog description by inserting a sound clip into the dialog description. For example, to play a sound after the stock quote, the producer inserts <resource type='SOUND' name='beep'/> right after the ADAPTER resource tag.

The producer can add a custom-made VXML script into the dialog description by using <resource type='RESOURCE' name='confirm'/> so that in the preferred embodiment, any VXML can be integrated into the dialog context transparently with maximum flexibility and expandability.

[0120] It will be apparent to one with skill in the art that while the example cited herein use VXML and XML as the mark-up languages and tags, it is noted herein that other suitable markup languages can be utilized in place of or integrated with the mentioned conventions without departing from the spirit and scope of the invention. It will also be apparent to the skilled artisan that while the initial description of the invention is made in terms of a voice application server having interface to a telephony server using generally HTTP requests and responses, it should be noted that the present invention can be practiced in any system that is capable of handling well-defined requests and responses across any distributed network.

[0121] FIGS. **7-15** illustrate various displayed Browser frames of a developer platform interface analogous to CL **141** of station **140** of FIG. **1**B. Description of the following interface frames and frame contents assumes existence of a desktop computer host analogous to station **140** of FIG. **1**B wherein interaction is enabled in HTTP request/response format as would be the case of developing over the Internet network for example. However, the following description should not limit the method and apparatus of the invention in any way as differing protocols, networks, interface designs and scope of operation can vary.

[0122] FIG. **7** is a plan view of a developer's frame containing a developer's login screen of **700** according to an embodiment of the present invention. Frame **700** is presented to a developer in the form of a Web browser container according to one embodiment of the invention. Commercial Web browsers are well known and any suitable Web browser will support the platform. Frame **700** has all of the traditional Web options associated with most Web browser frames including back, forward, Go, File, Edit, View, and so on. A navigation tool bar is visible in this example. Screen **710** is a login page. The developer may, in one embodiment, have a developer's account. In another case, more than one developer may share a single account. There are many possibilities.

[0123] Screen **710** has a field for inserting a login ID and a field for inserting a login personal identification number (PIN). Once login parameters are entered the developer submits the data by clicking on a button labeled Login. Screen **710** may be adapted for display on a desktop computer or any one of a number of other network capable devices following specified formats for display used on those particular devices.

[0124] FIG. **8** is a plan view of a developer's frame **800** containing a screen shot of a home page of the developer's platform interface of FIG. **7**. Frame **800** contains a sectioned screen comprising a welcome section **801**, a product identification section **802** and a navigation section **803** combined to fill the total screen or display area. A commercial name for a voice application developer's platform that is coined by the inventor is the name Fonelet. Navigation section **803** is provided to display on the "home page" and on subsequent frames of the software tool.

[0125] Navigation section **803** contains, reading from top to bottom, a plurality of useful links. Starting with a link to home followed by a link to an address book. A link for creating a new Fonelet (voice application) is labeled Create New. A link to "My" Fonelets is provided as well as a link to "Options". A standard Help link is illustrated along with a link to Logout. An additional "Options Menu" is the last illustrated link in section **803**. Section **803** may have additional links that are visible by scrolling down with the provided scroll bar traditional to the type of display of this example.

[0126] FIG. **9** is a plan view of a developer's frame **900** containing a screen shot of an address book **911** accessible through interaction with the option Address in section **803** of the previous frame of FIG. **8**. Screen **911** as an interactive option for listing individual contacts and for listing contact lists. A contact list is a list of voice application consumers and a single contact represents one consumer in this example. However, in other embodiments a single contact may mean more than one entity. Navigation screen **803** is displayed on the left of screen **911**. In this example, contacts are listed by First Name followed by Last Name, followed by a telephone number and an e-mail address. Other contact parameters may also be included or excluded without departing from the spirit and scope of the invention. For example the Web site of a contact may be listed and may also be the interface for receiving a voice application. To the left of the listed contacts are interactive selection boxes used for selection and configura-

tion purposes. Interactive options are displayed in the form of Web buttons and adapted to enable a developer to add or delete contacts.

[0127] FIG. **10** is a plan view of a developer's frame **1000** displaying a screen **1001** for creating a new voice application. Screen **1001** initiates creation of a new voice application termed a Fonelet by the inventor. A name field **1002** is provided in screen **1001** for inputting a name for the application. A description field **1003** is provided for the purpose of entering the applications description. A property section **1004** is illustrated and adapted to enable a developer to select from available options listed as Public, Persistent, and Shareable by clicking on the appropriate check boxes.

[0128] A Dialog Flow Setup section is provided and contains a dialog type section field **1005** and a subsequent field for selecting a contact or contact group **1006**. After the required information is correctly populated into the appropriate fields, a developer may "create" the dialog by clicking on an interactive option **1007** labeled Create.

[0129] FIG. **11** is a plan view of a developer's frame **1100** illustrating screen **1001** of FIG. **10** showing further options as a result of scrolling down. A calling schedule configuration section **1101** is illustrated and provides the interactive options of On Demand or Scheduled. As was previously described, selecting On Demand enables application deployment at the will of the developer while selecting scheduled initiates configuration for a scheduled deployment according to time/date parameters. A grouping of entry fields **1102** is provided for configuring Time Zone and Month of launch. A subsequent grouping of entry fields **1103** is provided for configuring the Day of Week and the Day of Month for the scheduled launch. A subsequent grouping of entry fields **1104** is provided for configuring the hour and minute of the scheduled launch. It is noted herein that the options enable a repetitive launch of the same application. Once the developer finishes specifying the voice application shell, he or she can click a Create Dialog button labeled Create to spawn an overlying browser window for dialog creation.

[0130] FIG. **12** is a screen shot of a dialog configuration window **1200** illustrating a dialog configuration page according to an embodiment of the invention. In this window a developer configures the first dialog that the voice application or Fonelet will link to. A dialog identification section **1201** is provided for the purpose of identifying and describing the dialog to be created. A text entry field for entering a dialog name and a text entry field for entering dialog description are provided. Within the dialog description field, an XML resource tag (not shown) is inserted which for example, may refer to a resource label machine code registered with a resource adapter within the application server analogous to adapter **113** and application server **110** described with reference to FIG. **1**B.

[0131] A section **1202** is provided within screen **1200** and adapted to enable a developer to configure for expected responses. In this case the type of dialog is a Radio Dialog. Section **1202** serves as the business rule logic control for multiple choice-like dialogs. Section **1202** contains a selection option for Response of Yes or No. It is noted herein that there may be more and different expected responses in addition to a simple yes or no response.

[0132] An adjacent section is provided within section **1202** for configuring any Follow-Up Action to occur as the result of an actual response to the dialog. For example, an option of selecting No Action is provided for each expected response of Yes and No. In the case of a follow-up action, an option for Connect is provided for each expected response. Adjacent to each illustrated Connect option, a Select field is provided for selecting a follow-up action, which may include fetching data.

[0133] A Send option is provided for enabling Send of the selected follow-up action including any embedded data. A follow-up action may be any type of configured response such as send a new radio dialog, send a machine repair request, and so on. A send to option and an associated select option is provided for identifying a recipient of a follow-up action and enabling automated send of the action to the recipient. For example, if a first dialog is a request for machine repair service sent to a plurality of internal repair technicians, then a follow-up might be to send the same dialog to the next available contact in the event the first contact refused to accept the job or was not available at the time of deployment.

[0134] In the above case, the dialog may propagate from contact to contact down a list until one of the contacts is available and chooses to interact with the dialog by accepting the job. A follow-up in this case may be to send a new dialog to the accepting contact detailing the parameters of which machine to repair including the diagnostic data of the problem and when the repair should take place. In this example, an option for showing details is provide for developer review purposes. Also interactive options for creating new or additional responses and for deleting existing responses from the system are provided. It is noted herein that once a dialog and dialog responses are created then they are reusable over the whole of the voice application and in any specified sequence in a voice application.

[0135] A section **1203** is provided within screen **1201** and adapted for handling Route-To Connection Exceptions. This section enables a developer to configure what to de in case of possible connection states experience in application deployment. For example, for a Caller Reject, Line Busy, or connection to Voice Mail there are options for No Action and for Redial illustrated. It is noted herein that there may be more Exceptions as well as Follow-up action types than are illustrated in this example without departing from the spirit and scope of the present invention.

[0136] A Send option is provided for each type of exception for re-sending the same or any other dialog that may be selected from an adjacent drop down menu. For example if the first dialog is a request for repair services and all of the initial contacts are busy for example, the dialog may be sent back around to all of the contacts until one becomes available by first moving to a next contact for send after each busy signal and then beginning at the top of the list again on re-dial. In this case John Doe represents a next recipient after a precious contact rejects the dialog, is bust, or re-directs to voice mail because of unavailability. Section **1203** is only enabled when the voice application is set to outbound. Once the first dialog is created and enabled by the developer then a second dialog may be created if desired by clicking on one of the available buttons labeled detail. Also provided are interactive buttons for Save Dialog, Save and Close, and Undo Changes.

[0137] FIG. **13** is a screen shot **1300** of dialog design panel **1200** of FIG. **12** illustrating progression of dialog state to a subsequent contact. The dialog state configured in the example of FIG. **12** is now transmitted from a contact listed in Route From to a contact listed in Route To in section **1301**, which is analogous to section **1201** of FIG. **12**. In this case, the

contacts involved are John Doe and Jane Doe. In this case, the dialog name and description are the same because the dialog is being re-used. The developer does not have to re-enter any of the dialog context. However, because each dialog has a unique relationship with a recipient the developer must configure the corresponding business rules.

[0138] Sections **1302** and **1303** of this example are analogous to sections **1202** and **1203** of the previous example of FIG. **12**. In this case if John Doe says no to the request for machine repair then the system carries out a bridge transfer to Jane Doe. In the case of exceptions, shown in Route-To Connection Exceptions region **1303**, all the events are directed to a redialing routine. In addition to inserting keywords such as "Yes" or "No" in the response field **1302**, the developer can create a custom thesaurus by clicking on a provided thesaurus icon not shown in this example. All the created vocabulary in a thesaurus can later be re-used throughout any voice applications the developer creates.

[0139] FIG. **14** is a screen shot of a thesaurus configuration window **1400** activated from the example of FIG. **13** according to a preferred embodiment. Thesaurus window **1400** has a section **1401** containing a field for labeling a vocabulary word and an associated field for listing synonyms for the labeled word. In this example, the word no is associated with probable responses no, nope, and the phrase "I can not make it". In this way voice recognition regimens can be trained in a personalized fashion to accommodate for varieties in a response that might carry a same meaning.

[0140] A vocabulary section **1402** is provided and adapted to list all of the created vocabulary words for a voice application and a selection mechanism (a selection bar in this case) for selecting one of the listed words. An option for creating a new word and synonym pair is also provided within section **1402**. A control panel section **1403** is provided within window **1400** and adapted with the controls Select From Thesaurus; Update Thesaurus; Delete From Thesaurus; and Exit Thesaurus.

[0141] FIG. **15** is a plan view of a developer's frame **1500** illustrating a screen **1502** for managing created modules according to an embodiment of the present invention.

[0142] After closing all dialog windows frame **1500** displays screen or page **1502** for module management options. Menu section **803** is again visible. Screen **1502** displays as a result of clicking on the option "My" or My Fonelet in frame **803**. Screen **1502** lists all voice applications that are already created and usable. In the list, each voice application has a check box adjacent thereto, which can be selected to change state of the particular application. A column labeled Status is provided within screen **1502** and located adjacent to the application list applications already created.

[0143] The Status column lists the changeable state of each voice application. Available status options include but are not limited to listed states of Inactive, Activated and Inbound. A column labeled Direct Access ID is provided adjacent to the Status column and is adapted to enable the developer to access a voice application directly through a voice interface in a PSTN network or in one embodiment from a DNT voice interface. In a PSTN embodiment, direct access ID capability serves as an extension of a central phone number. A next column labeled Action is provided adjacent to the direct access ID column and is adapted to enable a developer to select and apply a specific action regarding state of a voice application.

[0144] For example, assume that a developer has just finished the voice application identified as Field Support Center (FSC) listed at the top of the application identification list. Currently, the listed state of FSC is Inactive. The developer now activates the associated Action drop down menu and selects Activate to launch the application FSC on demand. In the case of a scheduled launch, the voice application is activated automatically according to the settings defined in the voice application shell.

[0145] As soon as the Activate command has been issued, the on-demand request is queued for dispatching through the system's outbound application server. For example, John Doe then receives a call originating from the voice application server (**110**) that asks if John wants to take the call. If John responds "Yes," the voice application is executed. The actual call flow follows:

[0146] System: "Hello John, you received a fonelet from Jim Doe, would you like to take this call?"

[0147] John: "Yes."

[0148] System: "Machine number 008 is broken, are you available to fix it?"

[0149] John: "No."

[0150] System: "Thanks for using fonelet. Goodbye!"

[0151] System: Terminate the connection with John, record the call flow to the data source, and spawn a new call to Jane Doe.

[0152] System: "Hello Jane, you received a fonelet from Jim Doe, would you like to take this call?"

[0153] Jane: "Yes."

[0154] System: "Machine number 008 is broken, are you available to fix it?"

[0155] Jane: "I cannot make it."

[0156] System: "Please wait while fonelet transfers you to Jeff Doe."

[0157] System: Carry out the bridge transfer between Jane Doe and Jeff Doe.

[0158] When the conversation is completed, terminate the connection with Jeff and record the call flow to the data source.

[0159] The default textual content of the voice application is being generated by the text-to-speech engine hosted on the telephony or DNT server. However, the voice application producer can access the voice portal through the PSTN or DNT server and record his/her voice over any existing prompts in the voice application.

[0160] It will be apparent to one with skill in the art the method and apparatus of the present invention may be practiced in conjunction with a CTI-enabled telephony environment wherein developer access to for application development is enabled through a client application running on a computerized station connected to a data network also having connectivity to the server spawning the application and telephony components. The method and apparatus of the invention may also be practiced in a system that is DNT-based wherein the telephony server and application server are both connected to a data network such as the well-known Internet network. There are applications for all mixes of communications environments including any suitable multi-tier system enabled for VXML and or other applicable mark-up languages that may serve similar purpose.

It will also be apparent to one with skill in the art that modeling voice applications including individual dialogs and responses enables any developer to create a limitless variety of voice application quickly by reusing existing objects in

modular fashion thereby enabling a wide range of useful applications from an existing store of objects.

Auto-Harvesting Web Data

[0161] In one embodiment of the present invention one or more Websites can be automatically harvested for data to be rendered by a VXML engine for generating a voice response accessible by users operating through a PSTN-based portal. Such an enhancement is described immediately below.

[0162] FIG. **16** is a block diagram illustrating the dialog transition flow of FIG. **6** enhanced for Web harvesting according to an embodiment of the present invention. Dialog controller **604** is enhanced in this embodiment to access and harvest data from an HTML, WML, or other data source such as would be the case of data hosted on a Website. An example scenario for this embodiment is that of a banking institution allowing all of its customers to access their Web site through a voice portal.

[0163] A Website **1600** is illustrated in this embodiment and is accessible to dialog controller **604** via a network access line **1601** illustrated herein as two directional lines of communication. The first line is labeled Store/Fetch/Input leading from controller **604** into site **1600**. The second (return) line is labeled Data Return/Source Field. The separately illustrated communication lines are intended to be analogous to a bi-directional Internet or other network access line. An internal data source (**602**) previously described with reference to FIG. **6** above is replaced in FIG. **16** by Website **1600** for explanatory purpose only. It should be noted that multiple data sources both internal to server **110** and external from server **110** could be simultaneously accessible to dialog controller **604**.

[0164] Website **1600** provides at least one electronic information page (Web page) that is formatted according to the existing rules for the mark-up language that is used for its creation and maintenance. Site **1600** may be one site hosting many information pages, some of which are inter-related and accessible through subsequent navigation actions. Controller **604** in this embodiment is enhanced for Website navigation at the direction of a user's voice inputs enabled by rule accessible by accessing rule engine **603**. A data template (not shown) is provided for use by dialog controller **604** to facilitate logical data population from site **1600**. Dialog controller **604** analyzes both Website source codes and data fields as return data and uses the information to generate a VXML page for rendering engine **111**.

[0165] It is noted herein that all of the security and access mechanisms used at the site for normal Internet access are inferred upon the customer so that the customer may be granted access by providing a voice rendering (response) containing the security access information. This enables the customer to keep the same security password and/or personal identification number (PIN) for voice transactions through a portal as well as for normal Web access to site **1600** from a network-connected computer.

[0166] FIG. **17** is a block diagram of the voice application distribution environment of FIG. **1**B illustrating added components for automated Web harvesting and data rendering according to an embodiment of the present invention. In this example, workstation **140** running client software **141** has direct access to a network server **1701** hosting the target Website **1600**. Access is provided by way of an Internet access line **1704**.

[0167] It is noted herein that there may be many servers **1701** as well as many hosted Websites of one or more pages in this embodiment without departing from the spirit and scope of the present invention. A database store **1702** is provided in this example and illustrated as connected to server **1701** for the purpose of storing data. Data store **1702** may be an optical storage, magnetic storage, a hard disk, or other forms suitable for storing data accessible online. In one embodiment, data store **1702** is a relational database management system (RDBMS) wherein a single access may involve one or more connected sub servers also storing data for access.

[0168] The configuration of client application **141**, workstation **140**, server **1702**, Website **1600**, and database **1702** connected by network **1704** enables Websites analogous to site **1600** to be culled or harvested. Application **141** can read and retrieve all of the default responses that exist for each HTML script or scripts of another mark-up language. These default responses are embedded into application logic **112** and VXML rendering engine **111**. Once the content of a Web page has been culled and used in client **141** to create the rendering, then VXML engine **111** can access the Website successfully in combination with application logic **112** and database/resource adaptor **113** by way of a separate access network **1703**. For example, if a user (not shown) accesses Website **1600** through voice portal **143** from receiving device **135** (telephone), then he or she would be voice prompted for a password to gain access to the site. Subsequently, a voice rendering of the data on the site accessed would be recited to him or her over telephone **135**.

[0169] Generally speaking, the development process for a voice portal would be the same as was described above with references to FIGS. **9**-**15** above. Some additional scripting or input of dialog is performed using client application **141**. Rather that requiring that the application developer populate all of the fields from scratch, or re-apply previously entered options, fields used by the business logic as discussed earlier in FIGS. **9** through **15** may be created from information harvested from site **1600** in this case. For that purpose, a software adapter (not shown) is added to client software **141** that allows it to communicate with Web site **1600** and harvest the information, both from the source code comprising fields and labels, etc. as well as from data parameters and data variables. It is noted herein that the process for data access, retrieval and voice rendering is essentially the same with respect to the processes of FIGS. **2**-**5** above except that a Website connection would be established before any other options are selected.

[0170] In one embodiment, provision of connection **1703** between server **110** and server **1701** enables the security environment practiced between communicating machines such a secure socket layer (SSL), firewall, etc to be applied in the created voice solution for a customer. On the analog side, the security is no different than that of a call-in line allowing banking services in terms of wiretap possibilities etc.

[0171] It will be apparent to one with skill in the art that the method and apparatus of the invention can be practiced in conjunction with the Internet, an Ethernet, or any other suitable networks. Markup languages supported include HTML, SHTML, WML, VHTML, XML, VXML and so on. In one embodiment, the Websites accessed may be accessed automatically wherein the password information for a user is kept at the site itself. There are many possible scenarios.

[0172] The method and apparatus of the invention should be afforded to broadest interpretation under examination in

view of the many possible embodiments and uses. The spirit and scope of the invention is limited only be the claims that follow.

What is claimed is:

1. A system for developing and deploying a voice application using Web-based data as source data over a communications network to one or more recipients, comprising:

a voice application server connected to a data network for storing and serving voice applications, the voice application server capable of accessing a network server and Website hosted therein;

a network communications server connected to the data network and to the communications network for routing the voice applications to their intended recipients;

a computer station connected to the data network having control access to at least the voice application server, the computer station capable of accessing the network server and Website hosted therein; and

a software application running on the computer station for creating applications and managing their states;

characterized in that a developer operating the software application from the computer station accesses the target server and Website and creates at least one template for holding Web data according to logical order of site construction and links the templates to voice applications created by object modeling and linking, whereupon according to a triggering event, the triggered voice application accesses the intended Website and retrieves refreshed data from the site into the template and renders the retrieved data as voice delivered to intended recipients.

* * * * *