# Voice Dialogs for Asterisk

Aysel Bembieva, Dmitry Namiot

*Abstract* — **This article describes the existing Interactive Voice Responses technologies for program ATE (Automatic Telephone Exchange) and provides their analysis in order to create more perfect technology.**

**The paper proposes a new extension for the voice menu organization, in which all shortcomings of the existing technologies would be removed.**

**The paper also presents a method to describe the voice menu to allow anybody without special knowledge to configure voice menu system for itself.**

***Keywords*—telephone exchange, XML, Asterisk.**

## I. Introduction

Nowadays business draws up its own rules: in the condition of contemporary speed of life, constant time deficit, human resources shortage companies must be able to pay attention to each client. First of all what important to companies are incoming calls from clients. But increasing numbers of clients makes secretaries fail to manage their job well, and the clients are lost. The increasing of stuff number is not the decision. State-of-the-art technologies offer to us more flexible solution: The Interactive Voice Menu (IVM) [1].

It is a system of prerecorded voice messages, which routs calls using information captured from user's input via tone dialing. Telephone voice menu systems effectively transform touch-tone telephones into computer input and output devices.

Menus are commonly used in voice mail systems, interactive voice response systems, and a range of other speech applications. A voice menu is similar to a graphical pull-down or pop-up menu [2]. It presents a set of choices to, and accepts input from, the user. Voice menu items are presented with recorded or synthesized speech rather than in textual form. Technically input could be gathered from a speech recognizer or telephone keypad (instead of a mouse, keyboard or touch screen in the traditional systems). In our development for this paper we deal with telephone keypad.

There are a lot of companies that create the IVM systems for anyone. They do it for money. And if there is a necessity to modify already created menu, another sum must be paid.

The programmatic interface to a voice menu typically includes standard elements. At the first hand, it is a list of voice prompts. Voice prompt could be a text string that represents voice prompts for use with a text-to-speech synthesizer, or a file name for a digital playback system. These prompts are then used to present options (alternatives) to the user. Of course, the whole picture could be more complex. Our voice menu could handle lists of second (third) round prompts, etc. Our voice system could provide some help facility that should be described too, etc.

Since the custom menu items are subject to frequent updates, the creation of the tools is very important. Here came the need to extend this technology to allow anyone to create the Interactive Voice Response (IVR) [3] he wants for himself.

This article describes the analysis of existing technologies and creation the best one for the Asterisk platform [4]. Asterisk is an open source, converged telephony platform. Asterisk combines more than 100 years of telephony knowledge into a robust suite of tightly integrated telecommunications applications [4]. Asterisk contains out of the box a huge set of applications, such as voicemail, hosted conferencing, call queuing and agents, music on hold, etc. We have deployed Asterisk for a number of projects, such as Home Gateway [5], telecom mashups [6, 7] and Smart Home projects [8]. Our paper [9] presents a new development tool for developing Asterisk services as ordinary CGI scripts. And this article is one more our tool to facilitate launch and support Asterisk applications. It is a tool that simplifies the creation of voice menus.

## II. Related works

In order to find the ways of realization, let us refer to the internal Asterisk structure and explore the interaction mechanisms with the other systems. Internal platform's configuration mechanisms are transparent.

The change of Asterisk settings can be done by the changing of its configuration files. One of the most important configuration files is *extension.conf*, which defines the dial plan. The dial plan is the formal description of the routing scheme and calls processing. It directs each call from its source to the destination via different applications. All calls: voice mail, conference, auto attendant's menu or just phone call – are defined by dial plan's logic and concept.

In Asterisk the dial plan is defined as the list of the applications and their arguments, executed in a concrete order. Execution order is defined by the priorities. Each step is registered as follows:

*exten => <exten>,<priority>,<application>, [(<args>)]*

Article received Sept 20, 2013.

A.Bembieva – graduated from a master's program at Lomonosov Moscow State University.

D.Namiot is senior researcher at Open Information Technologies Lab, Lomonosov Moscow State University

The content of the *extensions.conf* file is divided into sections, called contexts. Each context contains defined static settings and definitions or executable dial plan commands. In other words, each context is a set of extensions with its own unique name.

Contexts are used in order to perform main functions of Automatic Telephone Exchange [10] (hereinafter referred to as ATE): security, callings routing, auto attendant, multilevel voice menu, authorization, callback, macros.

The platform flexibility, its openness, accessibility allow creation of new possibilities, one of which is voice menu. Actually, the dial plan (dial plan configuration) is the simplest form of voice menu. The dial plan configuration changing – this is exactly on what the first approach for voice menu creation is based.

After main sections all the rest of the file *extensions.conf* is devoted to a definition of the dial plan, where the IVR logic can be described. Let us see an example:

```
[sip-in]
exten => 444,1,Goto(menu,s,1)
 [menu]
exten => s,1,Set(home="/home/menu")
exten => s,2,Wait(1)
exten => s,3,Playback(${home}/welcome)
exten => s,4,Playback(${home}/menu)
exten => s,5,WaitExten()
exten => 1,1,Playback(${home}/dept)
exten => 1,2,Goto(s,1)
exten => 2,1,Dial(SIP/accounting)
exten => t,1,Playback(make_choice)
exten => t,2,Goto(s,1)
exten => i,1,Playback(wrong_choice)
exten => i,2,Goto(s,1)
```

In spite of the simplicity, this approach has at least one shortcoming: any change in IVR each time affects the dial plan. So, there is a need to reload Asterisk configuration in order the changes will take effect. Obviously, this approach is not considered to be acceptable.

Another possibility that we want to investigate - is the integration of external scripts. Asterisk supports external scripts intercommunication interface – AGI: Asterisk Gateway Interface. AGI is Asterisk's analogue for CGI scripts in web programming. AGI is integrated with Asterisk and represents a method for execution of external scripts. Scripts can extend Asterisk functionality by means of other programming languages, such as Perl, PHP, C, Java, etc. AGI scripts interact with Asterisk via standard input and output streams.

So, our second approach for voice menu implementation is external script. This script (AGI-application) would contain all IVR-related logic.

A huge benefit of this approach is that there is no need to know all aspects of work with Asterisk configuration files. As a minimum, we can offer our own syntax for menu. The main disadvantage of this approach is the necessity to rewrite our script each time the IVR is changed. It means that the next step is almost obvious. Let us use the generic external script (it is not changes) and describe IVR itself as an external data source for the above mentioned script. In other words, this approach consists in the connecting of the AGI script and the external service that would provide the service.

One of the main ways of voice service organizing is VoiceXML It is the W3C's standard XML format for specifying interactive voice dialogues between a human and a computer [11].

The principal part in the system is the VoiceXML Interpreter, which is responsible for original code recognition. Context monitoring module - VXML Interpreter Context – performs operations for interpretation VXML document, finds out and serves incoming calls to a number which is linked to a VXML-script.

Web server stores VXML scripts set. Each script is mapped to a telephone number. At every call to a predefined number web server transmits the realization of a definite script to the VoiceXML Interpreter. Implementation Platform implements functions concerned with the work of the VXML script, e.g. produce or recognize speech via special servers: TTS (Text-To-Speech) and ASR (Automatic Speech Recognition), plays audio, etc.

VXML syntax is very similar to HTML. Each menu item is enclosed in the tag <form/>. Each menu item can contain <audio/> tag with the name of the file to be played for the chosen menu item. There are also the following tags:

- <noinput/> - event handler: if there were no entered input,
- <nomatch/> - event handler: if entered data doesn't correspond to any from the existing variants.
- <filled/> - event handler: if user entered correct data, next steps are made according to the scenario from this tag.

There are some shortcomings interfering with the visualization of the scenario: all menu items, no matter whether it a definite item –endpoint or just a submenu – are listed sequentially. But of course VXML eliminates the need to study the configuration for Asterisk.

Our idea was to combine the simplicity and portability of VXML with the efficiency of direct update configuration files.

## III. Proposed extension

Our solution is based on a new extension for Asterisk, for which the following aspects must be actual:

- This extension would use internal platform mechanisms for the efficiency.

• It would cover shortcomings of all technologies listed earlier.

The voice menu structure format must be simple for both program interpretation and human understanding.

There are some integration mechanisms for a new extension, but they are different according to extension type. Asterisk allows creating extensions with the following types: channel drivers, dial plan applications and functions, resources, codecs, etc.

The extension for the voice menu will deal mostly with the dial plan, so it pertains to the dial plan applications. They provide the whole system with the main functionality for calling. Applications can answer the calls, play the audio files, etc. Their call is scripted in the dial plan. E.g.:

```
exten => 6123,1,MyApplication(one,two,three)
```

Similar to VXML, we will describe our menu in the external XML file. Let us see an example (real service for Lomonosov Moscow State University):

```
<menuItemList> <!—-MSU_Welcome.mp3-->
  <item digit="1"> <!—CMC.mp3-->
    <menuItemList>
      <item digit="1">
        <endpoint>
          <file>11-Dean.mp3</file>
          <abonent>SIP/211</abonent>
        </endpoint>
      </item>
      <item digit="2">
        <endpoint>
          <file>12-Accounting_dpt.mp3</file>
          <abonent>SIP/212</abonent>
        </endpoint>
      </item>
    </menuItemList>
  </item>
  <item digit="2"> <!—Mech-Math.mp3-->
…
</menuItemList>
```

This example demonstrates that the main tag is <menuItemList/>. It contains information about menu items choosing for each level. On the first level there is some welcome message (just a media file to be played). Further there are options described in <Item/> tag, and «*digit*» attribute is for the key to be pressed. In other words, this structure lets quickly automate the following dialogue: "To reach the faculty of the Computational mathematics and Cybernetics press 1", "For the mechanic and mathematic faculty press 2", etc.

Each menu item can represent itself a menu for another items set. It is unlike VXML, where all possible submenus were located on the same level. So, <Item/> tag can contain <menuItemList/> tag description too.

E.g., pressing "1" causes switching to voice menu for CMC faculty with its own set of menu items, where for each item can be played own greeting: "To reach the Dean's office press "1"", "To reach the Accounting dpt. Press "2", etc.

If menu item means switching to the final telephone subscriber, <Item/> tag must contain <endpoint/> tag unlike VXML where event handlers were defined.

Tag <endpoint/> definitely specifies the final (the real) telephone subscriber by the tags: <abonent/>, <file/>.

There is also a possibility to define service settings similar to <noinput/>, <nomatch/>, but unlike VXML, there they can be defined only once, without any duplicates. Also if it will be necessary this option can be turned off.

```
<config>
  <SoundConfig name="Hello" enabled="1">
    <file>Hello.mp3</file>
  </SoundConfig>
  <SoundConfig name="SelectBranch" enabled="1">
    <file>SelectBranch.mp3</file>
    <repeat>no</repeat>
  </SoundConfig>
  <SoundConfig name="Timeout" enabled="1">
    <file>Timeout.mp3</file>
  </SoundConfig>
  <SoundConfig name="ErrorOnRedirect" enabled="1">
    <file>ErrorOnRedirect.mp3</file>
  </SoundConfig>
....
<config/>
```

Following options were picked out to be read as standard service messages:

• The possibility to specify the standard greeting message ("Hello, …");

• The possibility to specify the selection prompt message (SelectBranch);

• Standard timeout message (Timeout);

• Message to inform about call redirection error (ErrorOnRedirect);

• Informational message about call redirection (Redirecting);

• Possibility to specify the timeout for switching to default number (ShortTimeout), and exiting (LongTimeout).

Asterisk IVR extension can be called from a function, defined in dial plan configuration file. Voice menu description file must be given as the input.

```
exten => 999, 1, ivm("config_path");
```

Calling to this number (it is 999 in our example), causes menu settings reading from "config_path" via the EXPAT library. Having used the library calls, configuration file data is being interpreted and kept as a structure, written in C. So, internal structure CONFIG_STRUCT contains all information about configuration menu file:

```
typedef struct _CONFIG_STRUCT
{
  PLAYFILE_STRUCT pf_hello;
  PLAYFILE_STRUCT pf_select;
```

```
    PLAYFILE_STRUCT pf_redirecting;
    PLAYFILE_STRUCT pf_sorry;
    PLAYFILE_STRUCT pf_timeout;
    PLAYFILE_STRUCT pf_internalError;
    BOOL bRepeatAllOnTimeout;
    MENULEVEL_STRUCT menuLevel;
    ENDPOINT_STRUCT* defaultEndpoint;
    DWORD dwShortTimeout;
    DWORD dwLongTimeout;
  } CONFIG_STRUCT;
```

Actually, the full source file can be downloaded from Google Code site: *http://code.google.com/p/my-ivm-system/source/browse/* .

Then there are played the standard greeting messages, messages offering menu selection prompt, if the corresponding options from configuration file are switched on. Then the system plays menu items. The implementation uses the following internal call:

*ast_streamfile(struct ast_channel *chan, const char *filename, const char *preflang )*

After voiced menu items, the system switches to wait for input data. There is the internal function

*ast_waitfordigit (long timeout)*

The global timeout for the input is defined via configuration file. After this time, the menu is sounded again sounded.

After the user's input was entered, there must be conducted the next menu configuration analysis and defined whether the entered data definitely identifies the final telephone subscriber. If it is so, call redirection must take place. Otherwise, such menu item would be considered as another submenu. In this case system goes into menu items playing procedure and user's input waiting again.

Default phone number in the configuration file lets us support the use cases where user didn't entered input at all. For this case a special message about the redirection to the default number can be played. If there wasn't defined such default number, the timeout message must be played before the disconnection.

And call redirection message ill be played for all redirected calls.

The deployment is pretty standard. The downloaded extension should be copied into directory *"*\asterisk\apps\"*. This directory is home place for all modules of dial plan application type. If the platform is already installed, it is necessary to rebuild Asterisk with the new extension. Then there must be module registration procedure via internal Asterisk commands.

In order to make this extension work for a selected number, it is necessary to add to the dial plan configuration file the following line:

exten => PHONE_NUMBER,1,IVR("PATH_TO_IVR.XML").

So, calling to a number *PHONE_NUMBE*R will make it work.

## REFERENCES

[1] Schumacher, Robert M., Mary L. Hardzinski, and Amy L. Schwartz. "Increasing the usability of interactive voice response systems: Research and guidelines for phone-based interfaces." Human Factors: The Journal of the Human Factors and Ergonomics Society 37.2 (1995): 251-264.

[2] Arons, Barry. "The design of audio servers and toolkits for supporting speech in the user interface." Journal of the American Voice I/O Society 9 (1991): 27-41.

[3] Corkrey, Ross, and Lynne Parkinson. "Interactive voice response: review of studies 1989–2000." Behavior Research Methods, Instruments, & Computers 34.3 (2002): 342-353.

[4] Van Meggelen, Jim, Jared Smith, and Leif Madsen. Asterisk: The Future of Telephony: The Future of Telephony. O'Reilly, 2009.

[5] Schneps-Schneppe, Manfred, and Dmitry Namiot. "About Home Gateway Mashups." International Journal of Open Information Technologies 1.5 (2013): 1-5.

[6] Schneps-Schneppe, M., & Namiot, D. (2008). Telco Enabled Social Networking: Russian Experience. In BALTIC CONFERENCE (p. 33).

[7] Schneps-Schneppe, M., & Namiot, D. (2008). Telco enabled social networking exampled by GEO tagging. APPLIED INFORMATION AND COMMUNICATION TECHNOLOGIES, 50.

[8] Schneps-Schneppe, M., Maximenko, A., Namiot, D., & Malov, D. (2012, October). Wired Smart Home: energy metering, security, and emergency issues. In Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2012 4th International Congress on (pp. 405-410). IEEE. DOI: 10.1109/ICUMT.2012.6459700

[9] Schneps-Schneppe, Manfred, Dmitry Namiot, and Andrey Ustinov. "A Telco Enabled Social Networking and Knowledge Sharing." International Journal of Open Information Technologies 1.6 (2013): 1-4.

[10] Todorov, P., & Poryazov, S. (1985). Basic Dependences Characterizing a Model of an Idealised (Standard) Subscriber Automatic Telephone Exchange. Proc. of the 11-th ITC, 4-3.

[11] Farley, K. M., O'Reilly, J., Squire, L., & Farley, M. (2002). Voice application development with VoiceXML. Sams.

[12] Namiot, D., & Sneps-Sneppe, M. (2013, March). Wireless Networks Sensors and Social Streams. In Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on (pp. 413-418). IEEE. DOI: 10.1109/WAINA.2013.27