ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 *2006. Nr. 5(69)*
ELEKTRONIKA IR ELEKTROTECHNIKA

TELEKOMUNIKACIJŲ INŽINERIJA
T 180
TELECOMMUNICATIONS ENGINEERING

Review of Voice Dialogues in Telecommunications

A. Rudžionis, K. Ratkevičius, R. Maskeliūnas
Speech Research Laboratory, Kaunas University of Technology
Studentų str. 65, LT-51369, Kaunas, Lithuania; phone: +370 37 354191; e-mail: alrud@mmlab.ktu.lt
V. Rudžionis
Dept.of Informatics, Kaunas Humanities Faculty of Vilnius University
Muitinės str. 8, LT-44280 Kaunas, Lithuania; phone: +370 37 354191; e-mail: vyrud@mmlab.ktu.lt

Introduction

Speech processing is an important technology for enhanced computing because it provides a natural and intuitive interface for the user. People communicate with one another through conversation, so it is comfortable and efficient to use the same method for communication with computers. Voice technologies – speech recognition, text-to-speech, and speaker verification – are now mature enough to create a vital mode of customer contact, equally powerful as live agents and the Web. They have the potential to dramatically reduce the number of routine inquiries and transactions handled by agents and boost customer satisfaction by offering easy-to-use, always-available access from any landline or mobile phone.

Speech technology is also the future technology of e-business because it enables more natural, intuitive, and engaging customer service for less cost. The numerous benefits of speech technology for e-business include:

- Interaction with callers is easier and more natural;
- Menus can be eliminated or flattened, for more subtle and intuitive navigation;
- Call durations can be minimized, meaning less cost per transaction;
- Interaction with customers can occur 24 hours a day, 7 days a week;
- Customers interact with your business using their telephone or cellular telephone, resulting in continuous access to the customer base regardless of their location;
- Individuals with physical or perceptual disabilities might have greater access to e-business services;
- Enterprise branding is extended to a new channel – phone-based interaction;
- Overall customer service costs are decreased;
- Return on investment for speech application development often occurs in as few as three to six month;
- Opportunities for integrating and streamlining business processes arise as speech applications are developed;
- New business opportunities can arise using speech technology.

An effective dialogue is the key component to a successful interaction between a voice-only (telephony) application and a user. A voice-only application interacts with the user entirely without visual cues. The dialogue flow must be intuitive and natural enough to simulate two humans conversing. It must also provide a user with enough contextual and supporting information to understand the next action step at any point in the application. Because multimodal applications feature a graphical user interface (GUI) with which users interact, developers do not design dialogs for them. A hands-free application is an exception to this rule. Hands-free applications contain both a GUI and dialog-flow components, and provide users with both verbal and visual confirmations. A dashboard navigation system in a car is an example of a hands-free application. A user speaks to the application and the application speaks to the user, and a visual cue appears on a map, based on the user's input.

There are some alternatives for the developing and deploying of speech-enabled telephony applications:

- CAPI (*Common ISDN (Integrated Services Digital Network) Application Programming Interface*);
- Telephony API (TAPI)+Speech API (SAPI);
- Voice Server + Software Development Kit (SDK) + markup language.

The voice based timetable for long distance buses was created using CAPI: the user collects the known phone number and listens to directions by voice from computer, selects the departure town and the arrival town by voice, computer by phone presents some routes reading prerecorded speech phrases [1]. This approach is efficient

for telephony but is not well-suited to speech and internet applications [2].

Second approach integrates together telephony and speech. The SAPI application programming interface (API) dramatically reduces the code overhead required for an application to use speech recognition and text-to-speech, making speech technology more accessible and robust for a wide range of applications. The SAPI API provides a high-level interface between an application and speech engines. The two basic types of SAPI engines are text-to-speech (TTS) systems and speech recognizers. SAPI includes the speech grammar compiler tool, which enables to design voice dialogues in XML grammar format without changing the program source code. Microsoft's Telephony API (TAPI) provides developers with a standardized interface to rich selection telephony hardware. By utilizing TAPI, developers can write applications that support any device with a TAPI driver, also called a *Telephony Service Provider* (TSP). TAPI eliminates the need for developers to wrestle with device specific APIs and enables well-behaved device sharing between TAPI applications. Unfortunately TAPI is very complex and does not include direct support for useful speech technologies like text-to-speech (TTS) and speech recognition (SR). Now rather many companies offer TAPI controls (a collection of ActiveX and VCL (*Visual Component Library*) controls) [3], which you can call from your telephony application. These controls release you from the drudgery of writing low-level code.

Third approach integrates together telephony, speech and internet. So far it has only two kits to develop realizations: *Microsoft Speech Server* (MSS) and *IBM WebSphere Voice Server* [4]. The *IBM WebShpere Voice Server* is a VoiceXML 2.0 (*Voice eXtensible Markup Language*) -enabled speech environment. The VoiceXML is aimed at developing telephony-based applications, and takes the advantages of Web-based applications delivery to IVR (*Interactive Voice Response*) applications. Being different from IBM, Microsoft is using SALT 1.0 (*Speech Application Language Tags*) within *Microsoft Speech Server*. SALT targets speech-enabled applications across all devices such as telephones, PDAs, tablet PCs, and desktop PCs [4]. The *Microsoft Speech Application SDK* (SASDK), version 1.0 enables developers to create two basic types of applications: telephony (voice-only) and multimodal (text, voice, and visual) [5]. Run from within the Visual Studio.NET environment, the SASDK is used to create Web-based applications only. The SASDK makes it easy for developers to utilize speech technology. Graphical interfaces and drag-and-drop capabilities mask all the complexities behind the curtain. All the .NET developer needs to know about speech recognition is how to interpret the resulting confidence score.

Basic voice dialogue components

Real world users are unpredictable, and the system design needs to accommodate the wide variety of dialogs that may occur. For every node in the conversation, the designer needs to employ a consistent strategy or set of behaviors that can gracefully accommodate the range of unpredictable turns the conversation may take. Every voice dialogue could be composed of a collection of recognition and non-recognition states:

- Recognition states prompt the user for some input;
- Non-recognition states are effectively output only.

Since a non-recognition (or output) state does not require input from the user, error handling is generally not an issue. Further we will focus on the basic strategies most commonly used to handle errors and commands within a recognition state.

Typically, a recognition state has a main or initial prompt that is played on entry, plus a number of other supporting prompts that either restates the question or directive in a contextually appropriate way or offers help as the user traverses the state. Fig. 1 shows basic voice dialogue components [5]. Successful recognitions proceed to the next state. One of the most common schemes in use today involves:

- A main or initial prompt;
- A first silence prompt;
- A second silence prompt;
- A first mumble prompt;
- A second mumble prompt;
- A help prompt.

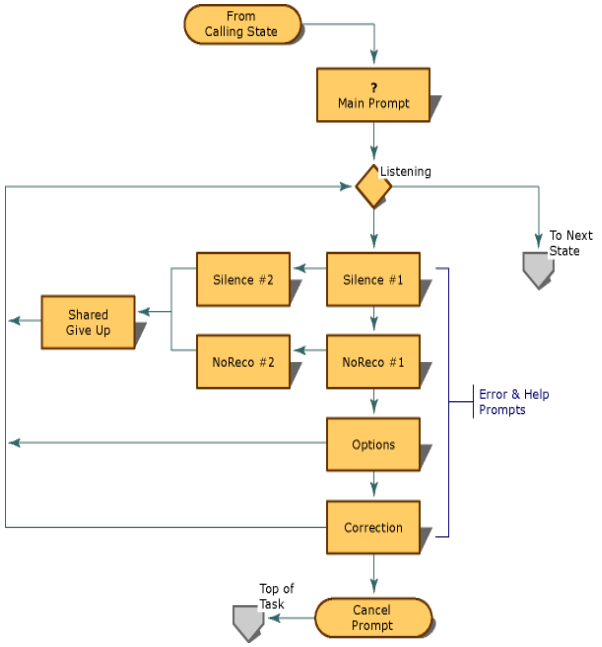

Fig. 1. Basic voice dialogue components

Other common elements employed in some systems are:

- A success prompt;
- A cancel prompt;
- A status prompt;
- A repeat prompt;
- A correction prompt.

Main or initial prompt is the default or entry prompt for the dialogue. Generally, this prompt is played on first

entry. All other prompts in the state support this prompt. In some cases a design may call for a tracked entry. Uses may vary, but a tracked entry for a main prompt is usually used to track first time entries, rather than subsequent entries, to a dialogue and adjust the prompts accordingly. For example, if a system's design allows users to loop back to a dialogue where the initial prompt was "Who should I invite?" the tracked entry prompt might say "Who else should I invite?" on all subsequent entries.

Silence prompt: by tracking the number of times the system does not hear anything from the caller, specific prompts can address the error in ways that are appropriate to the context. Usually, there are two levels of silence time-outs followed by a give-up prompt.

No recognition or mumble prompt: by specifying additional variables to store the number of misrecognitions, the system is able to track how successfully the user is negotiating the dialogue and respond with increasingly specific help messages. The maximum value for these prompts is at the discretion of the system designer. Currently, two is the norm.

A give-up or failure prompt can either send the user back to some predetermined state in the system to try another approach, or the system may offer to connect the user to a live operator.

Help prompt is a context-sensitive prompt that is triggered when the user invokes the system's help command. The help command is usually global in scope and is therefore generally available anywhere in the system.

The cancel prompt acknowledges the user's request to exit the state. Typically, when exiting a state because of a cancellation request from the user, the system will return to some task-appropriate signpost or landmark. The cancel prompt can also discourage users from exiting a multi-step interaction that they have nearly completed.

Features of voice dialogues used in SASDK

We will shortly overlook the specific features of voice dialogue organization with Microsoft's SASDK. There are two possible styles of voice dialogues in SASDK:

System-Initiative dialogue style. Using the system-initiative style, a sequence of specific questions or prompts guides a user through an application. The application asks the user a question, and accepts only an answer to that specific question. Dialogue occurs sequentially. Each question and answer cycle consists of one question and one answer. System-initiative dialogues are typically simpler to design than those using mixed initiative, but they limit the amount of flexibility a user has when answering questions.

Mixed Initiative dialogue style. Using the mixed initiative style, a user can answer multiple questions at once. The application can accept an answer in response to a specific question, but it can also accept extra answers that apply to questions the application has not yet asked. This style enables non-sequential dialogue. Each question and answer cycle includes one question, and one or more answers. Mixed initiative dialogues are typically more difficult to design than system-initiative dialogues, but they provide users with greater flexibility when answering questions. Mixed initiative dialogues simulate human interaction more closely than system-initiative dialogues.

Usage of semantic information. The application needs to gather semantic information from the user's responses, and to retain that semantic information for use throughout the application. Designers can specify a confirmation threshold for a semantic item. The semantic item accepts data returned with scores in excess of the confirmation threshold. In this case, the semantic item's state property is set to *Confirmed*. Any data with scores below the confirmation threshold will be marked as needing confirmation. By setting both confirmation and rejection thresholds the designer can control the flow of the voice dialogue: the recognizer either accepts the incoming data, marks it as needing further confirmation or simply rejects it as out of grammar.

Confirmation strategies used in SASDK:

Yes/No confirmations. A Yes/No confirmation is one in which the user is asked to explicitly confirm an item or chunk of information gathered during a previous part of the conversation.

Short Time-out confirmation. Another confirmation strategy involves the use of a short time-out. The system includes the most recently gathered information as part of the next turn. In the absence of user denial, the system takes the user's silence as a tacit confirmation of the data.

Menus and lists. Menus provide users with a list of choices. Because of the linear nature of speech, the number of items in a menu should be limited. Callers will not be able to hold more than three or four options in mind at one time. Grouping choices in some memorable way can increase the usable number to four or five.

There are three types of lists in SASDK:

- Coasting Lists;
- User-driven Lists;
- Numbered Lists.

Coasting lists are the audio equivalent of a slide show. Typically, a header introduces the list and tells the listener how many items to expect. In most cases, each item begins with marker or signposting phrases that explain their place in the list to the user.

User-driven lists rely on navigational commands from the user to advance from one item to the next. Typically, user-driven lists also begin with a description of the number of items on the list followed by a summary of the first item.

Numbered lists present a group of items, each preceded by a number. This scheme is suited to items that may share identical titles or labels.

Examples of speech-enabled Web and telephony applications

Speech application "Form filling by voice" was created to demonstrate Lithuanian voice dialog possibilities and could be used in such areas, as internet banking, e-shops, data acquisition and registration systems, etc. Speech can be implemented in two ways:

using *"Voice Web Studio"* toolkit for *"Macromedia Dreamweaver MX"* [2] as regular speech-enabled HTML webpage or using Microsoft's SASDK as telephony speech-enabled web application.

Scenario of voice dialogue: computer (either via computer speakers or telephone) greets user and asks to what company he would like to make the transaction (possible answers: shops "Minima" or "Maxima"). After user's response computer shows the recognized input. In case of incorrect recognition or silence, computer asks to repeat (giving possible answers) or to speak up more loudly (e.g., "I'm sorry I can't hear you, could you speak up more loudly please"), otherwise executes the second part of the voice dialog asking user how much he would like to pay (e.g., one or two Litas). Once all required information is gathered, program asks for confirmation (e.g., "would you like to transfer one Litas to Maxima's account?").

Multimodal version of speech-enabled Web application was made with "Voice Web Studio" (Fig. 2). It allow the user to choose the appropriate input method, whether speech or traditional Web controls. Main window contains regular HTML form elements: text display areas (text area) and text input fields (text field). Two main SALT elements (listen and prompt) were used for speech input and output. This sample is hosted on Speech Research Lab's website (http://www.speech.itpi.ktu.lt/ /demo/eb/default.html).

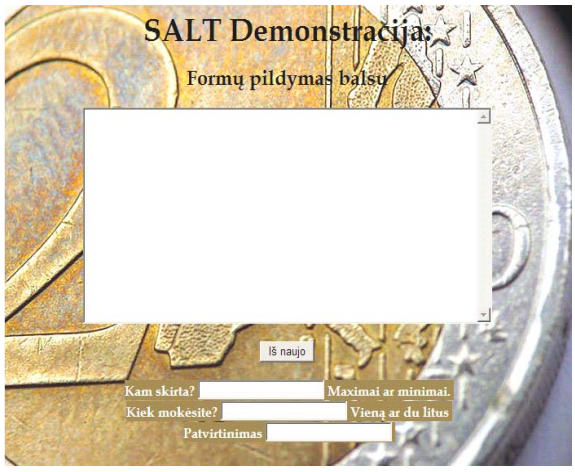

Fig. 2. Multimodal speech-enabled Web application "Form filling by voice"

Speech-enabled "voice only" telephony application with the same scenario was developed with SASDK. In this section we'll describe main steps of this application creation. First step is opening of *Visual Studio* and creation of *Speech Web Application* project. Second step is buildup of grammar rules. Grammar is a representation of everything the user can be expected to say, with certain selections tagged with semantic properties and values. Grammars are built in grammar editor using *List* and *Phrase* (recognizable words) elements (Fig. 3). Each rule in the grammar specifies semantic information in *Semantic Markup Language*. From the *Grammar Toolbox*, a *Script Tag* element must be dragged and dropped in the *List* element just to the right of the *Phrase* element with the text "minima". Developer must assign property "*Return result in a sub-property of this Rule*" in *Semantic Script Editor*.

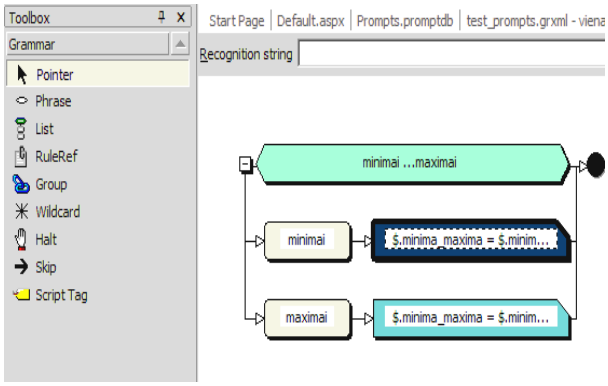

Fig. 3. Grammar editor

Third step is creation of dialog framework (Fig. 4). *Speech QA* (Question-Answer) controls prompt user for information and can also recognize user answers. Properties on the *QA* control also assign information to semantic items for use in responding to the recognized responses. All *QA* controls, except welcoming *LabasQA1*, must be dragged into *ASPX* panel in order to control them with *QASpeechControlSettings1*. *AnwerCall1* and *SemanticMap* controls are inserted by default.

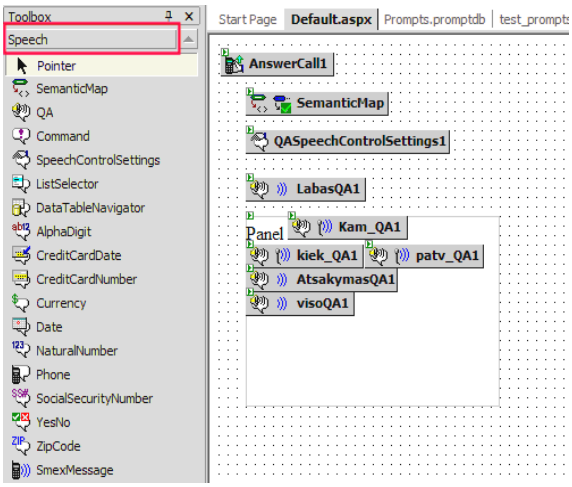

Fig. 4. Dialog framework

Next step is to record voice prompts. All the prompts are kept in a prompt database. Prompt transcriptions and recordings are done with *Speech Prompt Editor*. Developer must create extractions (bookmarks corresponding to recorded file) by placing brackets ([]) around the word or full phrase in *Transcription* field. When all transcripts and extractions are ready, it is possible to record the audio data for each transcription. Recording is done with *Recording Tool*. Speech

recognition engine processes recorded audio data and creates alignments (marks the end of each word in the audio data). After successful processing, a small wave icon asterisk appears in the *Has Wave* column and a green check mark appears in the *Has Alignments* column (Fig. 5). If the green check mark does not appear, the alignment failed (developer must re-record that phrase or word).

Speech recognition engine processes user input and returns semantic information to *SemanicMap* control. The *SemanticMap* control specifies semantic items to be used throughout the application to contain the semantic information. In the *Properties* window of *SemanticMap* control, developer must create three *SemanticItem* elements: *siminimai_maximai, siviena_du, siTaip_Ne*, specify recognition rules (by pointing to correct grammar file and selecting active rule) in every *QA* control (except welcoming *LabasQA1*) and indicate which recognition results get bound to which *SemanticItem* elements.

In order to make voice dialog more effective, application should confirm the responses that it has recognized to ensure that it has not recognized some phrases incorrectly, and to give user a chance to correct the error. To play back the user's responses as a prompt, it is necessary to create a prompt function, which extracts the text from the semantic items that were filled with user's answers to previous questions. Developer must specify parameter name, validation value (on which specified event should occur) and runtime value in *Prompt Function Editor* (Fig. 6). In our application such variables as *spatvirtinimas* (to play back phrase "You chose"), *satsisveikinimas* (to play back phrase "Thanks, call again") and *sprompt* (fully constructed confirmation prompt i.e., "You chose" + user answer "yes" or "no" + "Thanks, call again") were used.

This telephony application could be heard at Speech Research Laboratory after connecting of Intel Dialogic board D41JCT to telephone line.

Conclusions

Voice server with SDK combines Web technology with speech-processing services and telephony capabilities in a single system. So far it has only two realizations: *Microsoft Speech Server (MSS)* and *IBM WebSphere Voice Server*.

An effective dialogue is the key component to a successful interaction between a voice-only (telephony) application and a user, so the strong requirements to the voice dialogue structure should be implemented. The *Microsoft Speech Application SDK* (SASDK) provides efficient means for the design of voice dialogues.

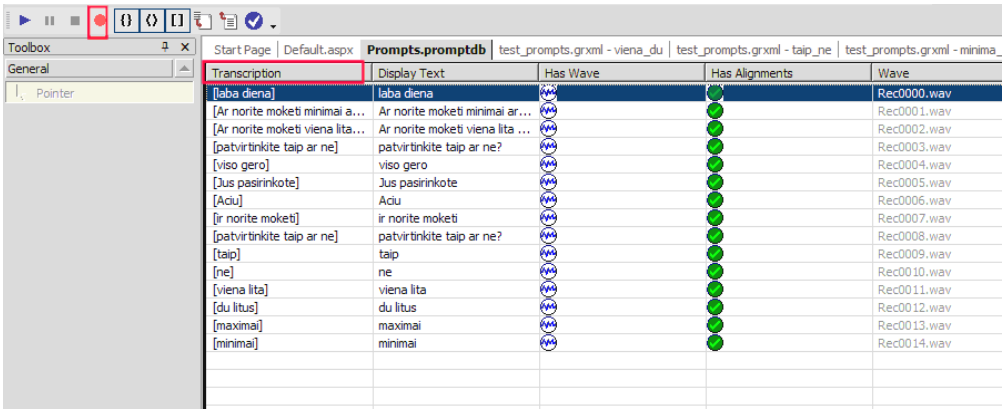

Fig. 5. Speech Prompt editor

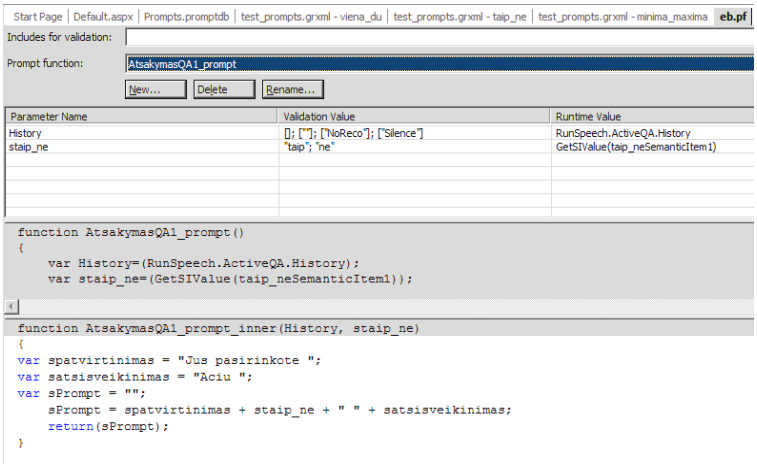

Fig. 6. Prompt Function editor

Two versions of speech-enabled Web application "Form filling by voice" were created: multimodal version of this application is hosted on Speech Research Lab's website (www.speech.itpi.ktu.lt/demo/eb/default.html), "voice only" telephony application with the same scenario could be heard at Speech Research Laboratory (Studentu 65-108).

References

1. **Rudžionis A., Ratkevičius K., Rudžionis V., Kasparaitis P.** Voice operated informative telecom services // Electronics and Electrical Engineering. – Kaunas: Technologija. – 2003. – Nr. 3(45). – P. 17–22.
2. **Rudžionis A., Ratkevičius K., Rudžionis V.** Voice based internet services // Electronics and Electrical Engineering. – Kaunas: Technologija. – 2004. – Nr. 3(51).– P.5-9.
3. **TAPI** custom controls. Retrieved April 27, 2006, from http://home.comcast.net/~bpennypacker/tapifaq/tapifaq4.html
4. **Rudžionis A., Ratkevičius K., Rudžionis V.** Speech in Call and Web Centers // Electronics and Electrical Engineering. – Kaunas: Technologija. – 2005. – Nr. 3(59).– P.58–63.
5. **Microsoft** Speech Application SDK. Retrieved April 27, 2006, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/SASDK_Tutorial/html/tut_introduction.asp.

Submitted for publication 2006 02 28

A. Rudžionis, K. Ratkevičius, R. Maskeliūnas, V. Rudžionis. Review of Voice Dialogues in Telecommunications // Electronics and Electrical Engineering. – Kaunas: Technologija, 2006. – No. 5(69). – P. 77–82.

Paper deals with the voice dialogue organization in speech-enabled Web pages with telephony access. Alternatives for the developing and deploying of speech-enabled telephony applications are reviewed. Basic components of voice dialogues are analyzed. Specific features of voice dialogue organization with *Microsoft Speech Application SDK* (SASDK) are presented. Speech application "Form filling by voice" was created to demonstrate Lithuanian voice dialog possibilities. Multimodal version of this application was made with *"Voice Web Studio"* and is hosted on Speech Research Lab's website (http://www.speech.itpi.ktu.lt/demo/eb/default.html). Speech-enabled "voice only" telephony application with the same scenario was developed with SASDK. Main creation steps of this application are presented. It could be heard at Speech Research Laboratory (Studentu str. 65–108). Ill. 6, bibl. 5 (in English; summaries in English, Russian and Lithuanian).

А. Руджёнис, К. Раткявичюс, Р. Маскялюнас, В. Руджёнис. Обзор голосовых диалогов в телекоммуникациях // Электроника и электротехника. – Каунас: Технология, 2006. - № 5(69). – С. *77–82*.

Анализируются принципы организации голосовых диалогов для интернетных приложений, к которым можно обращаться по телефону. Представлены альтернативы для создания голосовых диалогов для интернетных приложений и базовые компоненты голосовых диалогов. Также представлен набор средств для программирования голосовых диалогов, содержащийся в пакете *Microsoft Speech Application SDK* (SASDK). Описаны две версии интернетного сайта "Заполнение форм голосом": мультимодальная версия помещена по адресу http://www.speech.itpi.ktu.lt/demo/eb/default.html и телефонная версия, созданная с помощью SASDK, которая может быть прослушена в лаборатории исследования речи (ул. Студентов 65–108). Ил. 6, библ. 5 (на английском языке; рефераты на английском, русском и литовском яз.).

A. Rudžionis, K. Ratkevičius, R. Maskeliūnas, V. Rudžionis. Balsinių dialogų telekomunikacijose apžvalga // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2006. – No. 5(69). – P. 77–82.

Nagrinėjami balsinių dialogų organizavimo principai, skirti balsiniams tinklalapiams, kai vartotojas kreipiasi į tinklalapį telefonu ir nemato jokio vaizdo ekrane. Apžvelgiamos alternatyvos balsiniams dialogams kurti bei pagrindiniai balso dialogų komponentai. Pateikiamos balso dialogų kūrimo priemonės, esančios *Microsoft Speech Application SDK* (SASDK) pakete. Paruošti balso dialogą demonstruojantys tinklalapiai: įprastinis tinklalapis „Formų pildymas balsu", tik išplėstas balso sąsaja (prieiga per internetą http://www.speech.itpi.ktu.lt/demo/eb/default.html) ir to paties tinklalapio versija „tik balsas" (gali būti pademonstruota Kalbos signalų tyrimo mokslo laboratorijoje, Studentų g. 65–108). Trumpai apžvelgiama šio tinklalapio paruošimo, naudojant SASDK paketą, metodika. Il. 6, bibl. 5 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).

DOI: 10.5755/j02.eie.10682