Abstract

A complete development environment for designing, building and running voice operated services has been created. It offers a system builder a graphic platform with several types of blocks, such as an ASR block, a TTS one, a switch block, a database query block, etc. Even a large dialogue scheme can be realized in very short time simply by placing blocks on the form, specifying their properties and aligning them into meaningful dialogue branches. Sequencing the blocks into branches is solved in a unique way without using any interconnecting lines, which makes the dialogue scheme easy for editing. The platform, named LOTOS, has been employed in building a large multi-domain information system with voice access via telephone.

1. Introduction

Voice operated services become more and more popular both among telecom providers as well as their clients. The latter can benefit from a fast growing spectrum of interactive services that are available via (fixed or mobile network) virtually to anybody, at any place and any time [1].

Voice communication with a remote computer may have different forms, depending on available technology, on task definition and its difficulty and, last but not least also with respect to the target user group. While some narrow constrained services, such as, inquiries on weather forecast, phone banking or transport information may allow their users to say their demands in almost natural way, some service areas (and language communities, too) must rely on more or less formal way of interaction with the computer.

The latter approach is used in our system InfoCity, which was developed in 1998 and since that time it has been serving to public [2]. It is a multi-domain system, i.e. it offers a large selection of information from different sources, such as transport (train, coach, tram, bus) time-tables, culture and sport programs, opening times in many institutions, etc. It has been proposed as an open system, whose information portfolio could be easily extended and modified, which happens quite frequently. For a system like that a computer controlled scenario is necessary as the users need a sort of guidance when communicating with the system. (The guiding is also a way to overcome the troubles caused by Czech language, which has a very complex lexical, morphological and grammatical rules.)

The two-year experience with running, maintaining and innovating the InfoCity, together with increasing demand for services of this type, led us to a decision to create a universal platform for fast development of voice dialogue applications.

2. Components for voice dialogue systems

A designer creating a voice operated service needs to have at least the following modules: an automatic speech recognition (ASR) unit, a text-to-speech (TTS) unit, an information database with a structured access and a central unit managing the flow of the dialogue between the user and the computer. A system running over telephone further requests a module for handling single or multiple calls.

2.1. ASR module

Our ASR module employs hidden Markov models (HMM) to represent phonemes, words and word strings. There are 41 phonemes in Czech and each of them is modelled by 3-state 32-mixture context-independent HMMs. Signal cepstrum and its time derivatives make 20-dimensional acoustic feature vectors. The recent version of the ASR module is capable of classifying single words and word strings from a vocabulary with several thousands items [3]. Real-time continuous speech recognition is not possible yet, mainly due to the very complex nature of Czech language. With some limitations, however, a word spotting mode can be used to detect key-words in fluent speech.

2.2. TTS module

The TTS module provides direct translation of text messages, prompts and questions into a fairly intelligible speech output. It is also based on cepstral signal representation, which allows for sophisticated control of speech prosody. This module was developed in the Institute of Radioelectronics in Prague and it is the only system part made outside of our lab [4].

2.3. Dialogue manager

The applications developed so far were based on the system driven approach. At each dialogue level the computer offers multiple choices that are explicitly listed (if this is appropriate and the choice is not larger than some 5 options), or may be stated by a category name such as a week day, date, time, destination city name, etc. The user is prompted to respond by saying one of the indicated words or word strings. The recognition module matches the input with the currently active part of vocabulary that includes expected words, usually with multiple pronunciation variants and some frequently used synonyms, and several general control keywords that allow the user to change the standard dialogue flow (words like error, end, start, repeat, etc.). In the original version the manager was controlled by an external script written in a specially designed Dialogue Description Language.
3. Graphic environment for dialogue design

An alternative, graphic approach to building a dialogue system has been already proposed in paper [5]. A group of researchers from OGI developed the Rapid Application Development (RAD) system that allowed even a non-expert to create a simple voice operated application in an easy way. The RAD employed a small set of functional blocks that could be drawn on a screen and linked together to make a sequence of several dialogue turns. The system was originally aimed at fast prototyping of sample tasks but soon it found another application area as an educational tool in introductory courses on speech technology [6].

Inspired by the RAD we started to develop our own system for fast and easy design of voice operated applications. Our goal was to make a tool that would suit practical needs, like those occurring in the InfoCity service. We wanted the system to meet the following requests:

- complete visual design of computer-driven dialogue scenarios,
- design transparency even for large tasks,
- applicability in services providing automated access to multiple information databases,
- intuitive design, easy editing and modification without a knowledge of the system internals,
- environment similar to visual programming languages (toolbars, drag-and-drop interface, debug options),
- possibility to run the completed application within, as well as without, the development environment.

3.1. LOTOS layout

The layout of the system, which we named LOTOS, is depicted in Fig.1. Its largest part represents a drawing sheet prepared for a dialogue plan built from individual bricks. There are seven types of functional bricks available in the recent version. All are represented by their icons placed on the toolbar along the left margin. A brick is selected by clicking on its icon. This will cause a new instance of the brick being created and attached to the previously activated one. The recently added brick becomes active, which allows the designer to set its properties. The activation of an already placed brick is done by clicking on its box. As shown in Fig.1, no interconnection lines are needed. A link between the active output point and the input point of a new brick is established when the latter is added. To make the scheme compact and linear, only the path going to the active brick is displayed completely. All the other paths are hidden temporarily being just indicated by diminished shapes. This feature allows the designer to place even large plans on relatively small and compact space of the drawing sheet.

In the upper part of the design form there are icons for starting a new project, opening an existing one and saving the current one. Another set of icons allows the designer to specify variables, directories and source databases used during processing and managing the dialogue.

3.2. LOTOS bricks

A brick is the elementary unit in building dialogue schemes. It represents a single action of one (occasionally two) of the system modules, such as the ASR or TTS. Each brick is drawn as a box having an input, one or more output points, a label, a shortcut (indicating e.g. a TTS prompt) and a key to property definitions. In the following all bricks are described in detail.

3.2.1. TTS brick

Fig.2 shows a sample instance of the TTS brick. It has label Welcome and its only property is a message to be synthesised. The typed-in text may include application and system variables (such as %time%, %date%, %destination_city%) that are evaluated during the dialogue processing.

Figure 1: An example of a dialogue scenario built in the LOTOS environment.

Figure 2: TTS brick with its property box open.
3.2.2. ASR brick

The ASR brick represents a single action of the built-in recognition module. The module takes all the vocabulary items valid for the given brick instance and matches them with the input signal. If the brick has just one output the recognised item is stored in the variable specified by the designer. When recognition-conditioned branching is needed, the number of output points can be increased by clicking on the extension arrow of the branch bar. The property box allows for setting up echoing and key-word spotting options (Fig.3).

![Figure 3: ASR brick and its basic properties](image)

Words and word phrases to be recognized must be typed in the appropriate branches, as shown in Fig.4. After a new item is entered, its phonetic transcription is automatically derived and displayed. (It can be corrected manually if necessary.) Instead of typing the word list, the designer can open, use and edit previously created vocabulary files. The recognition module follows default actions in cases like, no speech occurs in the given time limit or classification is rejected. The default actions can be overridden by clicking on the two upper right icons and specifying the more appropriate response.

![Figure 4: Entering word lists to ASR brick branches](image)

3.2.3. Question brick

When analysing schemes of real (computer driven) dialogues we must notice that the most frequently occurring structure is a pair of system question followed by user answer. That is why the LOTOS has a special brick that includes functions of both the TTS and ASR modules (Fig.5). The question brick allows the designer to enter a question or a prompt and immediately set the parameters for processing the expected speech input. This combined block has all properties of the individual ASR and TTS bricks together with a new option that may allow a duplex mode and speaker barge-in. (The latter facility is still under development.)

![Figure 5: Question brick combines functions of TTS and ASR](image)

3.2.4. Switch brick

This block allows to branch dialogue flow according to the variables whose values were set in previous dialogue levels. For, example, the scenario must be split into two branches if the (travel, visit) day specified by the user is a work day or weekend, as depicted in Fig 6. The switch brick can evaluate also expressions that include more variables.

![Figure 6: Switch brick makes branching based on variables](image)

3.2.5. Jump brick

In case when the scheme cannot be arranged in the standard linear way, a jump brick must be used. It represents a dialogue move to the brick whose label is specified in the property box. This brick allows the designer to simplify schemes by creating loops, sharing parts of the scenario, admitting early exits or bypasses.

![Figure 7: Jump brick changes the standard linear flow](image)

3.2.6. Expression brick

This brick allows to add more complex decision taking into the dialogue scenario. It is capable of evaluating programs written in VB script that include application and system variables. Here, the designer must know at least the basics of VB script.

![Figure 8: Expression brick employs VB script facilities](image)
3.2.7. Time delay brick

Including a short pause might be useful if the designer wants to give the user some time, e.g. to perceive a longer list of data read from a database. Its only property is delay time in ms.

3.2.8. Database query brick

Picking up the requested piece of information from a database and reading it to the user is usually the most important action of the dialogue. In the LOTOS system the interaction with the data sources is accomplished by the query brick (Fig.9). The brick takes some keywords, usually the variables collected within the previous dialogue turns and intermediates a search for the appropriate database records. The valid records and their number are returned via an array of variables.

Moreover, the query brick can play an active role in the dialogue, such as providing data for variable parts of TTS prompts. The brick can also be included in early dialogue stages to check the number of records complying with the already known requests, which can determine, whether next dialogue turns are necessary or not. If, for example, there is just one coach connection in the specified day, it has no sense to ask the user which time he/she would like to depart.

![Figure 9: Query brick set for searching in train time-tables](image)

4. Working in the LOTOS environment

The LOTOS system has been developed for practical work, which must be fast, efficient and applicable for real services. Our experience show that an initial version of a desired dialogue scheme can be designed in time measured in tens of minutes rather than hours. This is supported by built-in default options that try to fit to the most usual situations.

In any moment, the designer can test the already made scheme by pressing the Run button and checking the live performance. During the run the system displays the dialogue path, highlights the activated bricks and monitors the recognition results. After encountering an error in the scheme, the designer simply stops the operation and immediately makes corrections or addition of new bricks. Our aim was to make the system friendly even for non-expert users. So, the easier the system looks, the more complex was its internal design. The LOTOS runs on a PC with the Windows98/2000 and supports both direct and telephone communication.

To test the system capabilities we employed it in redesigning the InfoCity service. (Just recall that this service provides voice access to 6 different databases each with its specific interface: coach, train and city transport time-tables, monthly sport and culture programs and other information like opening times.) For building the complete scheme we consumed 110 bricks, from which 65 were the combined question bricks. The scheme includes also a help branch with several levels and uses the active database query approach, which makes the dialogue more efficient for the user.

5. Conclusions

Two years after launching the first Czech voice-operated telephone service InfoCity we have developed a platform that should enable much faster and more user-friendly design of similar applications. Though our main domain is research in speech recognition, we are sure that working on this application oriented product did not mean wasting our time. On contrary, it helped us in re-shaping our research goals towards speech recognition paradigms, applicable to Czech, that contribute to more comfortable communication with computer systems. Soon, we would like to introduce a new LOTOS brick capable of accepting more than just one keyword per input by applying a more advanced word-spotting technique.

6. Acknowledgments

This work was partly supported by research projects of Grant Agency of the Czech Republic (no.102/96/K087) and by project MSM 242200001.

7. References