

Enabling Speech Based Access to Information Management Systems over Wireless Network

M. Bagein, O. Pietquin, C. Ris and G. Wilfart¹

Faculté Polytechnique de Mons - TCTS Lab.
Parc Initialis - Av. Copernic, 1, 7000 Mons - Belgium
{bagein, pietquin, ris, wilfart}@tcts.fpms.ac.be

Abstract

While wireless networks provide a greater autonomy to working people, interacting with a distant system is still constraining in terms of focus and abilities. Indeed, using a mobile terminal generally involves to look at it and use it manually. On another hand, last decade's improvements in the field of speech processing techniques have allowed to realize voice enabled interfaces which represent a more natural way to interact for human operators and free their visual focus as their hands. Yet, low cost and low consuming mobile devices are not suitable for the implementation of complete speech processing algorithms which are often CPU and memory consuming. Moreover, inherent problems of wireless networking, such as connection loss, have to be taken into account in the design of a voice enabled interface.

In this paper we propose an architecture of a mobile and distributed voice enabled interface for accessing to customer applications.

1 Introduction

In many domains, such as logistics, stock management or distribution centers, a human operator interacts with machines. This interaction is made via Human-Computer Interfaces (HCI), which allow users to communicate with an application server. Such domains often require operators to be mobile, for instance in warehouses where online stock management can be achieved by an instantaneous interaction between the operator and stock management applications, via the picking guidance interface. In the case of mail order selling companies or distribution centers, picking procedure can also be improved by a mobile interface, especially if the device is carried by the operator himself because of frequent manipulations of items with a wide variety of characteristics (pens, jewels, etc.). Besides mobile interfaces are attractive for the real-time and remote aspect of assisted maintenance tasks as they can prevent waste of time (online access to technical information) and create a more effective and efficient maintenance workforce with a direct update in the maintenance management application (stock update, reports, etc.).

¹This work is sponsored by the Walloon Region through the "Initiative" MODIVOC project.

An efficient HCI must then allow communications over a wireless network. In this environment, voice also appears as a natural way to interface with machines, adding extra advantages in terms of ergonomics, as hands and eyes remain free, and productivity by avoiding the necessary time of extra manipulations as form-filling, data encoding, etc., either manually (paper print out) or electronically (bar-code scanner, screens, keyboard, etc.). Our purpose is to realize a real interface over an existing application. So, it is acknowledged that no change to the existing application architecture is allowed. Only original interactions with the application are permitted and our goal is to ease those interactions by enabling voice access.

In this paper we discuss an original way to realize such an interface in two ways. Firstly, we propose a complete software solution that is supposed to work on any hardware architecture composed of fixed and mobile devices of any kind (laptop, PDA, custom device, etc.). Doing so, no particular hardware material is imposed to the customer and previously installed hardware architecture can be used where other existing solutions require to buy specific and non-reusable hardware. Secondly, we propose an evolving and easily upgradeable solution. Indeed, the proposed architecture takes into account large and non-static vocabularies in the opposite of other existing solutions. Moreover, as we propose a fully software solution, new speech processing technologies can easily be integrated without modifying the rest of the interface (hardware and network architecture).

In section 2, we expose the main constraints linked to the design of the proposed solution. In section 3, we propose a software architecture meeting those requirements. Then, we detail the interface in terms of dialogue management (section 4), Automatic Speech Recognition (ASR, section 5), and Text-To-Speech (TTS, section 6) systems.

2 Constraints

Such an interface imposes ergonomic, but also economic and technical constraints. In the rest of this section, we investigate the requirements of the system.

Ergonomic constraints: Mobile devices should not interfere with the operator's movements. They should be light, robust and low consuming. Those constraints mainly impose the use of embedded devices, that have typically low CPU and low memory capacities.

The interface itself should be as intuitive as possible, and possibly require no particular user training. In such a way, the design of the dialogue is a crucial point, as, if efficient, it can greatly improve the performance of the overall system in terms of operator's satisfaction.

Economic constraints: The minimal requirements for the mobile audio devices are a microphone, a headset and a wireless network adapter. Due to the possible use at large scale, mobile hardware should be low cost. In this sense, speech interface hardware is usually cheaper than its visual counterpart. This gives the voice interface another particular advantage.

On the other hand, this might imply low-quality acoustic devices (microphones, ...), certainly not dedicated to speech recognition. The poor quality of the acquisition must be taken into account by the speech recognition system.

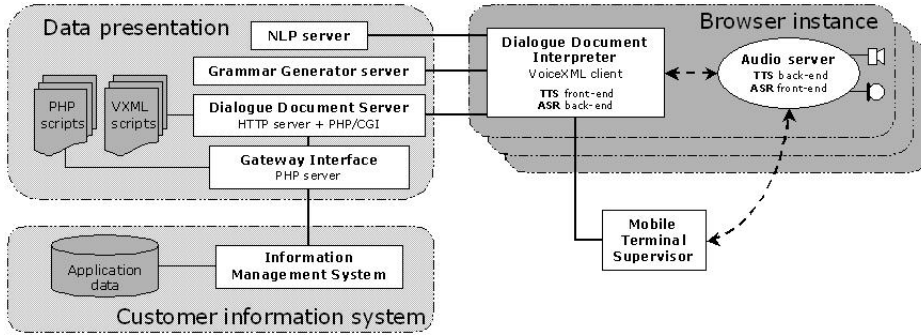


Figure 1: *Software architecture. Dotted arrows symbolize data streams over wireless network.*

Technical constraints: We propose a voice interface to existing applications, and thus do not want to change them. We need, however, to communicate with the applications. This implies that those applications rely on an information system we can access through a query mechanism. Typically, such an information system would be a relational database accessed via drivers supporting the SQL formalism.

Target applications also imply that the environment is often noisy. Thus, the speech recognition system should be robust enough to handle this problem.

Moreover, a natural voice HCI faces the complex problem of spontaneous speech recognition, namely handling hesitations, out of vocabulary words, etc. Once again, an efficient dialogue design can greatly improve the performance of the voice interface

Speech generation is also part of the interface. In the dialogue process, what needs to be uttered depends on the database content, and is not known in advance. Speech synthesis brings increased flexibility and easier upgrades of applications, as any text is converted into speech with no additional processing. It is therefore preferred over pre-recorded prompts.

On another hand, mobile devices are connected to the fixed terminals through a wireless network which introduces new constraints: due to the common medium of the radio waves (atmosphere), the unique network resource between mobile and fixed devices is limited, shared and not expandable. According to the radio conditions (distance, obstacles, reverberation, etc.), the available bandwidth can severely degrade. As concrete applications can require up to several hundreds of simultaneous communications, the use of efficient compression algorithms (in terms of bandwidth and CPU requirement) is essential.

3 Software Architecture

In our approach, we consider that the operator uses speech to interact with the customer information system.

In the next topics, we describe some details about the implementation of speech technologies (TTS, ASR and dialogue management) in a mobile environment and its integration in a classical 3-tiers software architecture (figure 1).

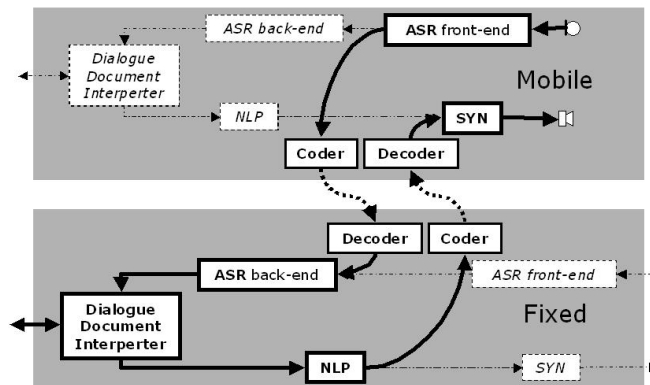


Figure 2: Algorithm distribution over wireless network. TTS (front-end=NLP, back-end=SYN) and ASR algorithms are illustrated with their respective codecs.

3.1 Browser instance

Embedded audio software: At the software level, the operator interacts essentially with the application through ASR and TTS systems but the possibility to record and play audio memos must be also available. Due to ergonomic and economic reasons, the embedded CPU and memory resources might be very limited in the mobile device. Then, the embedded software set cannot include all the components required by the browser instance.

One solution is to split the browser algorithms between mobile and fixed devices. In our approach, we consider that audio player, audio recorder, ASR front-end and TTS back-end are implemented in mobile devices (figure 2). The counterparts of ASR and TTS and the dialogue document client (Voice eXtensible Markup Language - VoiceXML [1] - interpreter) are implemented on fixed devices. Each mobile runs a single instance of the embedded TTS/ASR software and this instance is connected, through socket mechanisms over the wireless network, to a single instance of a VoiceXML client.

Dialogue document interpreter: The VoiceXML client is able to load and parse dialogue documents. According to the dialogue description, the VoiceXML client can start the audio output dataflow (audio playing or TTS) and the audio input dataflow (audio recording or ASR). The result of the audio input processing is then analyzed to start a new action: jump to another point of current dialogue, run JavaScript actions, load a new dialogue document, etc.

We consider two types of audio output : audio memos and sentences. Audio memos are directly sent to the mobile device but in case of sentences, a TTS system is required. It is possible to process sentences in a full TTS process and transmit the result as audio memos but it is more efficient in terms of bandwidth and CPU to split the TTS algorithms and transmit intermediate data over the wireless network. The TTS front-end acts as a layer using a translator (section 6) between sentence and intermediate data. Once again, an optimization is possible by using a single Natural Language Processing (NLP, section 6.2) resource shared by all TTS front-end algorithms.

In the case of user input, before performing speech recognition, the ASR process must be initialized with a recognition grammar (section 5.3). As the NLP server, a single grammar generator resource can be shared by all ASR back-end algorithms.

3.2 Mobile Terminal Supervisor

When a mobile becomes operational and when it enters in the radio coverage, (reaching wireless network dedicated to the vocal application), the mobile device must be linked to a new instance of VoiceXML client. The job of the Mobile Terminal Supervisor is to register each new audio server, create new VoiceXML client instances and link them to each other.

Then, the client requests the audio server characteristics, availability, and capabilities about data flows it can support. Besides useful actions (login/logout and authorization management), the Mobile Terminal Supervisor must periodically check the current status of each mobile and detect connection losses. Those status information are required to handle suspended dialogue mechanisms (section 4.2). An implementation based on the Network Element Control Protocol (NECP [3]) proposal is used to perform this management. This document provides suitable functionality descriptions.

3.3 Data presentation

The vocal scenarios of the application are described in a set of documents respecting the VoiceXML formalism. The goal of the dialogue document server is to diffuse those documents towards clients. A classical HTTP server can perform this task. Dialogue documents can be stored in a file system or can also be generated from information management system data. In this case, a gateway interface (PHP server) should be used to generate queries and format the results inside a VoiceXML template.

Moreover, both the NLP server and the grammar generator will be part of the data presentation as those resources are shared by all the browser instances.

3.4 Customer information system

One of the main advantages of this architecture is the possibility to add a vocal interface without re-designing the customer Information Management System (IMS). All communications with the IMS are made through the dialogue document server which can support well-known gateway interfaces (PHP, Perl CGI, batch scripts, ...). For example, an IMS based on a relational database system can be accessed through SQL queries. Those queries can be parameterized by recognition results and query results can be included in audio prompts. The behavior of the voice interface is fully described in the VXML and PHP scripts.

4 Dialogue Management

Voice enabled applications do not only involve speech recognition and speech synthesis. Indeed, some management of the interaction is necessary in order to achieve the goal of the conversational game. A spoken dialogue system can be formalized as a finite automaton that steps from state to state according to the results of the actions it performs in its environment (here, the human operator

and the IMS). Typically, actions are prompts to human operators, databases queries etc. Thus, designing an optimal dialogue strategy can be seen as the fact of creating a mapping between the dialogue state space and the action set that leads to an appropriate sequence of interactions allowing the achievement of the operator's goal. This remains a complex task and is the object of many recent researches [6]. In this paper we will only focus on some particular points involved by our solution.

4.1 Enhancing speech recognition performance

As said before, for some reasons such as noisy environment or low cost acquisition systems, the speech recognition process is made difficult and recognition errors can occur. In order to achieve anyway the operator's goal, confirmation mechanisms have to be implemented. According to the actual state in the dialogue and the confidence level of the last recognized command, two confirmation mechanisms may be used:

1. Implicit confirmation: the next system prompt refers to an unsure command but asks the operator for a new information. The user can then correct the problematic command or implicitly agree by providing the new information he has been asked for.
2. Explicit confirmation: in the case of very low confidence levels or after having tried unsuccessfully another mechanism, the system asks explicitly to confirm the previous command.

Of course, the use of the first mechanism is more ergonomic and has to be preferred.

4.2 Handling suspended dialogues

Mobile devices evolving in a wireless network area may go out of coverage. Moreover, human operators may be forced to stop their interaction with the system, because they are interrupted by a colleague for example. For these reasons, the dialogue manager must be able to recover a stopped, and thus incomplete, interaction. It is then necessary to keep trace of the dialogue history.

To do so, the system fills in a template all along the interaction. This template collects all the information obtained so far by the system and remains active while the dialogue is not terminated. It is a representation of the dialogue state. Nevertheless, being able to continue a suspended dialogue is not enough, several mechanisms have to be implemented. First, the system must be able to decide whether it has to remind the operator what the dialogue state is. Indeed, if the dialogue has been suspended for a long time, the operator may have forgotten the interaction's history. On another hand, this should not be systematic as a short interruption does not require reminding. Actually, systematic reminding alters the ergonomics. Second, the embedded part of the system must be able to warn the operator that it has detected a connection loss and prevent him to continue the interaction.

4.3 Replacing visual information

Due to the voice enabled aspect of the application, the user does not have visual information. In usual Graphic User Interfaces (GUI), in order to ensure the correctness of database queries, for example, the possible choices of a combo-box in a given state, can be constrained by partial queries based on choices that occurred in previous states. This kind of mechanism is not possible with vocal interfaces because it would imply to enumerate all the possible choices in an audio prompt. This would be very bad for ergonomics if the list is large. Thus the user should be allowed to build queries that potentially provide no result. In this case the dialogue management must warn and advise the user in order to build a correct query in a natural way.

In another hand, in the ergonomic point of view of an HCI, the user should not have to wait and if that occurs he should know why. In traditional GUI, it is the job of progression bars, dynamic mouse cursors or status bars to provide information about the status of the current process. As our application relies on an existing customer system that we should not modify, it is necessary to handle possible weaknesses of this system such as time consuming database queries.

In order to improve the ergonomics of a voice enabled interface, the dialogue strategy has to take timing constraints into account and should not allow long pauses between utterances. To do so, it is necessary to introduce new utterances in the dialogue that provide information about the system's state. For example, it is important to warn the operator that a database query is being processed when this processing lasts for a while (processing time is easily measurable). Thus, the user knows that the system is working and not down.

5 ASR System

Automatic speech recognition is the part of the vocal interface which aims at transcribing into text what has been pronounced by the user. Integrating this technology in a wireless network architecture imposes several additional technical constraints that justify the need of distributing the ASR processes on both the mobile and fixed terminals.

5.1 Distributed speech recognition (DSR)

The part of the processing which is achieved on each terminal and the actual implementation of these processes result from a trade-off between the computation power available on the mobile end (mobile phones, PDA, laptops, ...), the transmission rate over the wireless connection and the required recognition performance.

1. The computation and memory capacities of the mobile devices can be very constrained while some parts of a speech recognition system require a lot of CPU, limiting the part of the processing achieved on the mobile terminal.
2. In the framework of a wireless connection, the available band-width appears as a major criterion in the choice of the implemented technologies. The necessity to keep the bit rate as low as possible will also impose, in a certain way, the actual distribution of the tasks over the network.

3. The coding algorithms applied to the data in order to limit the bit rate will have some impact on the performance of the speech recognition system depending on the type of compression schemes. We know, for instance, that a coding performed on the sampled speech signal (cf. GSM codec), is much more detrimental for the ASR performance than a coding of the parameters derived from the acoustic analysis of the speech signal (cf. DSR codec), especially when the transmission conditions are getting worse [7].
4. The maintenance of the applications - database/models updates, ... - is another aspect of the problem that should be taken into account. Indeed, it is preferable to avoid, as much as possible, the maintenance operations on the mobile terminals as they might become very complex at large scale.

These problems have been widely discussed in the speech community. The international project AURORA [2] has been created in order to propose solutions and finally establish an international standard for DSR (recently published by the ETSI - European Telecom Standards Institute [5]).

The speech recognition processing can roughly be split into four steps: 1) the Voice Activity Detector (VAD) which isolates the useful speech signal from the signal flow acquired from the microphone, 2) the acoustic analysis which aims at extracting from the speech signal, a relevant and non redundant representation of its acoustical content, 3) the acoustic modeling which aims at classifying the acoustic features into language dependent acoustic units, and 4) the decoding that integrates the vocabulary and the grammar into a hidden Markov model structure in order to obtain the word sequence hypothesis [8].

The ETSI standard proposes to perform the voice activity detection and the acoustic analysis on the mobile device while the acoustic modeling and the decoding are achieved on the fixed terminal. This solution satisfies all the constraints listed above. Indeed, the acoustic analysis requires very low and fixed memory and CPU loads, the coding of the acoustic features is efficient and has almost no impact on the recognition performance (a bit rate of 4.8 kbits/s is proposed by the ETSI standard). Finally, acoustic feature extraction requires no external data such as databases or models, it is independent of the language and of the application so that the maintenance of the embedded software is reduced to the minimum, practically software upgrades.

5.2 Noise robustness

Different external parameters can alter the performance of the speech recognition systems. Some of them such as intra- or inter-speaker variability, co-articulation effects, ... can be efficiently handled by the acoustic models. However, the actual conditions in which the voice interface is used induce particular causes of disturbance that require a specific processing. This is the case of environment noise specific to each application (background conversation, warehouse, engines, ...) or convolutive noise, due to the characteristics of the possibly low quality of the voice acquisition devices such as embedded microphone. Noise robust algorithms must therefore be integrated at different level of the speech recognition process. In particular, noise reduction techniques can be applied to the acoustic analysis, as it is the case in the ETSI standard, given that such processing should not drastically increase the CPU load on the mobile device.

5.3 The grammar generator

An ASR system recognizes complete sentences or commands resulting from particular combinations of the vocabulary words. The set of all the combinations of words that an ASR system can recognize composes the grammar. Using a grammar improves the recognition, not only by limiting the number of possible sentences, but also by taking decisions on better-recognized parts of the speech, thus enabling error recovery and improving noise robustness.

A dialogue system makes use of several grammars, depending on the dialogue state. Therefore, it must be able to handle several active grammars at the same time and to switch between grammars. An ASR system must meet these requirements to conform to the voicexml formalism.

The role of the grammar generator is not only to generate grammars from the IMS data, but also 1) to organize them in some optimal way, in a formalism that is understood by the ASR system (the grammar compilation), 2) to ensure that the necessary grammars are available with a minimum latency for the user and 3) to ensure that grammars are up-to-date.

To satisfy the last two points, the grammar generator should be able to compile grammars fast enough, and to reuse already compiled grammars (cache mechanism) as long as they do not need to be updated. Another requirement of the system is that it should handle queries that return no result (see 4.3). Therefore, the ASR grammar should be flexible enough as to recognize items that are not available in the IMS, warning the user that the query returned no result.

To keep the grammars up-to-date, it is necessary to check for changes in the IMS, thus query its state. This means that the use of the vocal interface generates additional queries. This, together with the use of a cache, justifies that we use a single grammar generator acting as a server of VoiceXML clients.

6 TTS System

In many cases, vocal messages (information, questions, ...) can be sent to the user with pre-recorded audio prompts. This technique seems simple but, in fact, is not compliant at all due to the obligation of pre-recording all audio prompts. A Text-To-Speech system aims at transcribing human readable texts into speech sound signals. With such a system, audio prompts can be automatically generated from prompt texts avoiding audio recording. Moreover, if the TTS system is used online, new prompts based on the result of IMS queries can be generated on the fly. Then the evolution from day to day of the IMS does not require a vocal interface update.

6.1 TTS algorithm distribution

TTS systems need databases [4, 9]. The size of those databases, typically about 30 Mb, is not compatible with mobile device resources. Due to the limited embedded resources and wireless transmission rate, the TTS implementation is also split into two parts. The fixed part of the TTS is the NLP, which aims at translating human readable texts into phonetic transcriptions (including prosodic features). The embedded part is the speech synthesizer, which aims at translating phonetic transcriptions into audio stream. The bit-rate required to transmit phonetic transcriptions from the NLP to the speech synthesizer is lower than 1.5 kbits/s.

6.2 Natural Language Processing

The NLP requires a lot of memory and CPU resources. Nevertheless, a sentence and its phonetic transcription is very compact. Moreover, most of sentences stay unchanged in the day to day usage of the application. Then, retrieving transcriptions from previous sentences with a cache mechanism prior to submitting a sentence to the NLP process reduces drastically the memory and CPU resource requirements. Eventually, the set of unchanged sentences can be shared between all operators and this last point justifies to include a NLP server as a part of the Data Presentation.

7 Conclusions

In this paper, we discussed the problem of interfacing distant applications over wireless networks using voice enabled interfaces. Starting from a classical 3-tiers software architecture, and taking into account ergonomic, economic and technical constraints, we have proposed a particular design of the speech processing tools based on a distribution of the tasks between the fixed and mobile terminals. Some aspects particular to vocal interfaces, such as dialogue design and grammar management, have also been discussed. In order to encourage re-usability and extendability of those tools, we mostly based our implementations on standards.

References

- [1] VoiceXML - <http://www.w3.org/TR/2003/CR-voicexml20-20030220/>.
- [2] Aurora 2.0 - <http://www.elda.fr/proj/aurora2.html>.
- [3] Cerpa, A. et al., "NECP: The Network Element Control Protocol", Work in Progress, <http://www.circleud.org/~jelson/writings/>
- [4] T. Dutoit, "An Introduction to Text-To-Speech Synthesis", Kluwer Academic Publishers, Dordrecht, 1997.
- [5] ETSI - Distributed Speech Recognition
<http://www.etsi.org/frameset/home.htm?/technicalactiv/DSR/dsr.htm>
- [6] E. Levin, R. Pieraccini, W. Eckert, G. Di Fabbrizio, and S. Narayanan., "Spoken Language Dialogue: From Theory to Practice" in Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU),1999.
- [7] D. Pearce, "Enabling New Speech Driven Services for Mobile Devices: an Overview of the ETSI standards activities for Distributed Speech Recognition Front-Ends", in Proc. AVIOS 2000, San Jose, CA, May 2000. <http://www.etsi.org/frameset/home.htm?/technicalactiv/DSR/dsr.htm>
- [8] L.R. Rabiner, "An Introduction to Hidden Markov Models", in IEEE ASSP Magazine, pp. 4-16, January 1986.
- [9] P. A. Taylor, A. Black and R. Caley, "The Architecture of the Festival Speech Synthesis System", in The Third ESCA Workshop in Speech Synthesis, pp. 147-151, 1998.