Telephone Interface for the Email Service

Nuno Baptista, Rui Prior
Instituto de Telecomunicações — Universidade do Porto
Porto, Portugal
Email: rprior@dcc.fc.up.pt

Manuel E. Correia
CRACS — Universidade do Porto
Porto, Portugal
Email: mcc@dcc.fc.up.pt

***Abstract*—Several studies have demonstrated the advantages of using IVR (Interactive Voice Response) technologies[5], which allow the users to phone a computer and access services by pressing the appropriate touch-tone keys on their telephones. Because it offers substantial benefits in terms of cost and time efficiency, there is a continuous pursuit for increased automation using IVR. The real challenge lies in providing a user-friendly, yet cost-effective interface to users in order to improve their interaction with existing services, making them more usable and useful, providing an experience that fits their specific objectives and utilization contexts. This paper describes a method and system for providing customizable audio access to email messages kept in several IMAP (Internet Message Access Protocol) backstorages using an IVR application that takes full advantage of TTS (Text-To-Speech) software.**

***Keywords*-IVR; voice user interfaces; touch-tone menus**

I. Introduction

Common forms of verbal and written interactions are currently being replaced by other more efficient communication forms in the virtual world [11]. There are many practical and economic reasons that explain this current state of affairs [13]. By being able to dematerialize its presence in the virtual world, an individual is able to conduct more then one conversation at the same time and disseminate its virtual presence simultaneously in many different places, increasing his communication capabilities by orders or magnitude. This is a game changer for successful social and economical interactions. The widespread adoption of cheap broadband Internet services, based on high-speed data communications technologies like cable modems, DSL (Digital Subscriber Line) or Fiber to the Home, is catalyzing new forms of digital services and facilitating the rapid integration of the Internet in daily activities by businesses and ordinary users.

Computer-based VoIP (Voice over IP) services provide real cost savings, portability and improved network utilization, with the convergence of voice and data on the Internet [4]. VoIP is currently being integrated with applications in widespread use within business Intranets and innovative Internet services [20]. These newer forms of integration have attracted more users to this technology, since services that were either unavailable in the PSTN (Public Switched Telephone Service) or were too complex or expensive to deploy are currently being made widespread by VoIP and associated services. This trend is expected to increase as the technology matures and is further integrated into new and innovative data services, creating and promoting new forms of communication.

There is an ever increasing need for mobile access to data services [15]. This tendency is accelerating the development rate of mobile devices, where the traditional forms of input and output (keyboards, mice, and high-resolution displays) are cumbersome or impossible to use. The need for mobility is also a driving force behind the decrease in size of the devices we use to interact with Internet services. Functionality and ease-of-use are rather limited on small devices, in which designers are trying to include more and more features into rapidly shrinking packages. This further exacerbates the inadequacy of current input/output devices in mobile contexts, making the use of alternative forms of user interaction not only advisable, but required. Thus, rather than merely shrinking the size of traditional interface elements, new I/O modalities must be explored.

In some contexts, reliance on visual interaction is disadvantageous, and it is a real problem for visually-challenged users. Under these circumstances, human/computer interaction must rely on some other sense. Hearing is an obvious candidate for an text-centric applications. Voice is the most immediate and natural means of communication for Humans. Computer interfaces are no exception, and the use of voice for human/machine interactions is nowadays widely spread [3].

When discussing communication in the Internet, it is inevitable to consider email as one of the most useful and widespread services [1]. Its ubiquity is unparalleled, being deployed in virtually all popular desktop and mobile operating systems. Email, however, has almost exclusively been accessed by the means of standard GUI (Graphical User Interface) applications. This paper presents a prototype of a practical IVR (Interactive Voice Response) interface to the email service, a telephonic MUA (Mail User Agent) that takes advantage of TTS (Text-To-Speech) technologies for relaying textual email information in the form of speech in mobile contexts where visual access (reading) is impractical. Our goal is to further leverage the ubiquity of email by providing a more versatile access method for situations where people are fully engaged in critical tasks requiring continuous visual attention (such as driving) or vision cannot otherwise be used (e.g., by visually impaired people).

While speech offers some unique advantages and opportunities for computer interaction, the known cognitive limitations [18] of speech-based interfaces amplify the importance of usability in the development of speech applications. IVR usability design and re-engineering know-how ranges from

style guides for touch-tone IVRs [10] to comprehensive collections of best practices for IVR design [17], and also cover state-of-the-art speech-enabled IVRs [2]. Based on sound, an IVR system necessarily involves a serial model for content presentation. Items at the top of the list are very prominent, becoming progressively less prominent as we go down the list. This contrasts with web design where visual techniques, like bold text, can be used to make items stand out, irrespective of their order. Thus, as a complement to the email IVR interface, we developed a web configuration service that is used for the user pre-configuration of the touch-tone IVR email user agent.

There were previous approaches to speech-based email access, e.g., Emacspeak [16]. However, Emacspeak still requires a computer with a regular keyboard. Together with the concept of virtual folders, introduced in a program called View Mail[12], it was used as inspiration for our work.

The rest of the paper is organized as follows. Section II describes the design of the voice interface and the associated web-based customization interface. Section III describes some implementation details of our prototype. Section IV discusses the different TTS modules used in our system. Section V concludes the paper and provides some topics for further development of this work.

II. System Design

We have used the IMAP model for email access [6] as a guiding framework for the hierarchical folder-based structure of the developed IVR interface. Our system is based on two complementary applications, one is based on voice and the other is based on the web.

The web interface is used to configure the IVR system. The IVR has many parameters that need to be set and adjusted prior to its usage, and this can only be done effectively by the means of a more traditional web application. Parameters are divided in several configuration groups, namely virtual email folders and message filters specifications, security credentials for backstorage access, email language automatic detection, and acronyms expansions for a more fluid user hearing experience. Using the web to perform these configuration tasks reduces immensely the complexity of the IVR interface, keeping it simple by focusing exclusively into the mechanics of pragmatic text-to-speech email delivery.

As key points of our IVR email management system we would like to highlight:

- Virtual folders: Our system can pool messages from different IMAP folders from several different e-mail accounts at the same time. Selection email filters can then be applied to fit each user needs in the form of specially configure virtual email folders. These settings are performed in the web interface, allowing for a great simplification of the voice interface. The use of virtual folders as the main form of hierarchical email organization allows for a great simplification of the hierarchy of voice menus that needs to be navigated when the user interacts with the IVR to listen to his email.
- Security: In order to store user data preferences, we have used a MySQL database. As the database holds users passwords, we have employed security measures and cryptographic protocols to store, use and verify user credentials. Since there are two access passwords in the system, one for the voice interface and the other for the web interface, each password is used as a cipher key to encrypt the other, thus safeguarding data integrity. The web access is subject to some security concerns that will be detailed later.
- Automatic language detection: Since we can use TTS systems with support for different spoken languages, we have also implemented automatic email text language detection in order to convey the email message to the user in the appropriate spoken language.
- Email acronyms expansion: Email writing is an informal means of communication that frequently makes heavy usage of acronyms and abbreviations. We have implemented an acronym expansion based mechanism to make email speech more fluid and understandable. The currently recognizable acronym list can and will be increased in the future. Users can add or overload acronym expansion entries using a personal acronym list.

A. Voice Interface

The voice interface plays a central role in the system. Whether using voice menus or graphical menus, the presentation of the software to the user as a list of items is one of the most popular styles of interaction. Such structure favors the user's performance by helping him form a mental model that is in agreement with the actual system model.

The development of the voice interface was performed having in mind the following aspects:

- Time: Voice applications have in time their added value. Feedback should be brief but informative in order to save time and reduce the amount of information that the user must hold in memory.
- Instant feedback: In a well-designed prompt, it must be possible to interrupt the audio at any time to proceed with a selected action. This feedback greatly increases the quality of the users' experience, allowing them to situate themselves in the application without incurring excessive time penalties.
- Informative feedback: The existence of informative feedback in a closed circuit helps the user situate himself after triggering an action. Appropriate state indicators should also be presented when the system is running a potentially lengthy process.
- Hierarchy: Unlike linear solutions, a hierarchical voice interface allows the user to recognize the tasks instead of being forced to recall them, greatly reducing the cognitive load of the system. Such interface is relatively easy to shape mentally and easily navigable using the phone keypad. The fact that email is usually organized in a hierarchy of folders further enhances the analogy between a hierarchical voice interface and the real world.

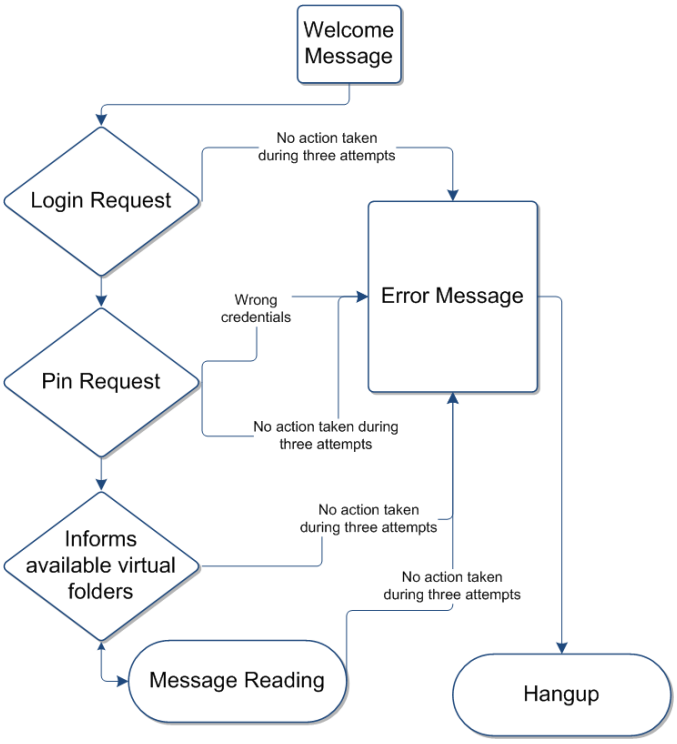

Figure 1. Voice interface

- Successive refinement: An iterative model, where in each iteration the circle is extended to increase the detail of the system, makes it easier for the user to locate a given item.
- Use of a consistent model: The use of a similar syntax for similar semantics avoids confusing the user, easing the act decision making, and allowing for a more gentle learning curve.

Interaction with the voice interface, illustrated in figure 1, is performed as follows. After presenting itself with a welcome message, the system asks the user for credentials. The username is typed using the T9 (text on nine keys) method [9], speeding up the process as each key needs only be pressed once. After entering the username, the user is asked to type a numerical password (PIN) for authentication. There is a timeout period for the interaction of both the username and the PIN, after which the request is repeated; if no action is taken after three timeouts, an error message is issued.

After proper authentication, the system loads configurations from the user profile, previously configured through the web interface and stored in a database. The profile contains, among other information, the selection and parameterization of the TTS system, the specification of the virtual folders, the configuration of the different email accounts and the personal list of acronyms for expansion (which is merged with the global list). Sensitive information is stored encrypted, keeping it safe in the event that the security of the database is compromised.

The user is then presented with a choice of the virtual folders from the profile. The system supports two different methods for the selection of virtual folders. In the first method, folder names are read to the user in groups of nine (should there be more than nine virtual folders configured in the profile); the user may select a folder at any time by pressing the corresponding key in the numeric keyboard. The * and # keys are used to move to the previous or next group of folders, respectively. By pressing 0, the user switches to the second method, direct folder name entry using T9. In order to save time, whenever a long enough prefix has been typed by the user so that the T9 hash prefix is unique (only one of the virtual folders of the user has a T9 hash beginning with the typed digits), that folder is selected without need for typing the entire folder name.

The virtual folder selection is followed by message reading. After informing the user of the number of messages contained in the virtual folder, the system reads the messages sequentially, sorted from newer to older. The fields of the message to be read are configured in the profile, the default being the subject and the message body. The user can interrupt the system at any time and move to the next message by pressing # or to the previous one by pressing *. Since message selection by direct entry using T9 would be next to useless, however, in message reading mode the 0 key is overloaded with a “back” function for returning to the virtual folder selection menu.

The voice menu system uses both pre-recorded and text-to-speech messages. Pre-recorded messages are used in the standard interaction items. This eliminates the need for repeatedly synthesizing common messages, avoiding conversion delays and lowering the system load. On-the-fly TTS conversion is performed whenever there is no standard interaction (e.g., choice of virtual folders or email reading).

B. Web Interface

The usefulness of the system is highly dependent on its capacity to lead the user to the interesting set of messages in the shortest possible time span. This capacity largely depends on the flexibility given to the user for specifying what messages are “interesting” and to categorize them into different interesting message sets—the virtual folders. The configuration of virtual folders and other settings in the profile is performed through a web-based interface provided by the system.

The web interface allows for configuration at two different levels: system-wide administrative level and ordinary user level. At the administrative level, it is used to create users and groups, set permissions, to visualize statistics regarding users’ access to both interfaces, and to manage the system-wide acronym expansion table. The ordinary user level is used for configuring each user’s profile. It provides a means for changing the password for the Web interface, set the PIN for authentication in the voice interface, select one of the available TTS systems and parameterize it, configure the different email accounts, define virtual folders and their respective filters, and manage the personal acronyms and abbreviations table.

The definition of virtual folders includes the selection of email accounts, the set of IMAP folders to include from each

account, and a set of filters for a fine-grained specification of the interesting message set. The filters can be based on message flags (answered, unanswered, seen and unseen), on message header fields (from, to, cc, bcc, subject, body and text), and also on the messages size and date/time. Complex filters can be created by combining multiple filters through the AND and OR clauses.

III. System Implementation

Our prototype of the system was implemented around a soft-PBX platform—Asterisk [21]. Asterisk is a complete open source PBX system that provides all of the features that would be expected from a PBX, including many advanced features often associated with high-end, costly proprietary PBXs.

One particularly interesting aspect of Asterisk is the possibility of extension, allowing for the implementation of functionality absent from the built-in feature set by means of external modules. Asterisk provides a native API (Application Programming Interface) for programming modules in C; it also provides the AGI (Asterisk Gateway Interface) for modules written in any other language. Inspired by the CGI (Common Gateway Interface) used in web servers to invoke language-agnostic scripts for generating dynamic web content, the AGI uses the standard I/O streams (stdin, stdout and stderr) for communication between Asterisk and external modules. The modules themselves do not need to know anything about the physical interface, protocol or codec of the call they are dealing with, since all of these details are abstracted out by Asterisk, which greatly simplifies the development of new telephonic services. AGI scripts launched by Asterisk can control the telephony operations on the associated control and voice channels; they are typically used to add advanced logic and to communicate with relational databases or access other external resources (as is the case of the email system).

Our prototype of the system was implemented in Perl. Perl is a stable, high-level, general-purpose, dynamic, multiplatform programming language, used in critical applications in several sectors, and very popular in Web development. It has a very large selection of modules available, including the Asterisk::AGI module for writing Asterisk extensions and several modules for accessing the email service and parsing the messages. The widespread use of HTML and other non-plain-text encoding methods in email messages makes the integration with the TTS system difficult. The outstanding pattern matching capabilities of Perl make it ideal for post-processing the message contents and feeding clean text to the TTS system.

A. Voice Interface

The voice interface was developed as an AGI module in Perl. AGI scripts communicate with Asterisk by sending AGI commands on the standard output and receiving responses on the standard input, typically an HTTP-like response code (200 for success, 5xx for errors).

Interaction with the email system was implemented using the Mail::IMAPClient module, which provides the methods for implementing an IMAP protocol client, allowing Perl scripts to interact with IMAP message stores. This module allows the access to multiple IMAP accounts, and provides support for message filtering, two necessary features for the implementation of the virtual folders. Secure email access is provided by SSL (Secure Sockets Layer) or TLS (Transport Layer Security), implemented by the IO::Socket::SSL module.

When an email message is retrieved for reading in the voice interface, the message body is flattened and the MIME::Parser module is invoked to parse the contained MIME streams. Email messages are frequently composed of multiple parts, and it is necessary to parse them in order to present only the relevant ones to the user; message parts irrelevant to the voice interface (e.g., a jpeg photo) are ignored. At this stage, the message parts are processed in order to generate the text to be read, ensuring, for example, that two plain text parts are concatenated, but if two alternative parts are present (e.g., HTML and plain text versions of the message) only one of them is read; processing is also needed to extract only the text from HTML parts. The final step before passing the text to the TTS system is acronym expansion, performed using pattern matching and substitution.

One small aspect that had a huge impact in the usability of the system was the pipelining of the (potentially lengthy) tasks of processing the email message and converting it to speech, fed to Asterisk in the form of compressed sound data. Standard Unix (anonymous) pipes were used wherever possible, and temporary named pipes wherever not. The SAPI TTS (see sec. IV), however, is incompatible with any standard form of pipelining, since it writes an empty header for the sound data and later `lseek()`s back to fill it. With this TTS converter, the solution was to write a program that simulates a pipeline using a temporary file but initially waits for the filesize to be larger than the header, meaning that the header has already been filled-in and actual sound data is being written. With this scheme, the initial delay was brought down from about 20s for an average-sized email and longer for larger emails to about 1s and mostly independent of the size of the email.

B. Web Interface

The web interface was implemented as a CGI script with a MySQL database backend. MySQL is an open-source RDBMS (Relational DataBase Management System) that provides high performance transaction processing and compliance with the ACID properties (Atomicity, Consistency, Isolation and Durability). The interface between Perl and the database is provided by the DBI module; the CGI Perl module deals with of fill-in forms and the details of state maintenance.

The creation of new user is made both in the MySQL database and in the asterisk users configuration files (in order to create an entry for VoIP access). This is achieved through communication established between the Web interface and a server script through a protocol based on the client-server architecture. This protocol uses shared key cryptography to avoid undue access. Using a server script enables the development of other modules to interact, for instance, with LDAP or database

systems, without implying changes in the code of the Web interface, as long as the specified protocol is respected.

In terms of security, since the email account configurations must be stored at the server (thus not under the control of the users) for use by the voice interface module, it is fundamental to ensure that no sensitive information (namely passwords) is stored in clear-text. Since the session IDs are stored on the server side and cookies are stored on the client-side, the following fields are stored in the database to achieve secure profile storage:

- Random user number: With a size of 512 bits, it is generated when the user is created and never leaves the database. It is generated using Crypt::Random (a Perl module to interface the /dev/random device).
- Password encrypted with PIN: The PIN used to access the voice interface acts as a key to encrypt the web interface password; encryption is done using the Crypt::OpenSSL::AES module, based on the Rijndael Algorithm.
- PIN encrypted with Password: The password used to access the web interface acts as a key to encrypt the PIN used in the voice interface.
- MD5 password hash: Used for authentication with the web interface; when the user is created, the PIN has not yet been defined, so an alternative authentication method is necessary so that users can enter the application.
- Password encrypted with an MD5 hash of (Session ID | random (the same as in the database) | salt (the same present in the cookie))

The session ID is a random number generated by the server. The cookie, stored at the client side (browser) and sent to the server with each request, contains the session ID and an additional random number generated each time a user logs in, the Salt. Even though the scheme is somewhat complex, it allows information to be accessible when it is needed, using a combination of server-side and client-side information, but never either of them alone. Security in communication is achieved through the use of https.

The web interface for administrators can also show statistics of the utilization of the voice and web interfaces by each user; these charts are produced with the GD::Graph::bars module.

IV. TTS Conversion

One key component of a voice interface is the TTS conversion system. Its quality (or lack thereof) has a huge impact on the usability of the interface. The quality of TTS conversion depends on two important factors: intelligibility (the ease with which the output is understood by humans) and naturalness (the closeness of the synthesized speech to actual human speech).

In this work, we have evaluated three different TTS systems—Mbrola, Espeak and the Nuance/Real Speak commercial voices using SAPI (Speech Application Programming Interface)—all of which support multiple languages. Even though the TTS system to be used can be selected through the web interface, our preference as the primary TTS system goes to the last one, due to its superior performance in the Portuguese language both in terms of intelligibility and naturalness.

TTS systems can be categorized into three families: concatenation synthesis, articulatory synthesis and formant synthesis. Most TTS systems use concatenation synthesis [14]. This synthesis processes a small quantity of data achieving good results with an acceptable computational cost. The lack of emotional speech features present in most of the strategies of this kind of synthesis limits their usefulness to the task of neutral speech [8]. Articulatory synthesis uses a model that mimics the mechanisms for producing human speech. It includes a model for the glottis, vocal cords, tongue and lips [7]. While articulatory synthesis is quite promising, it has received less attention compared with the other synthesis methods and did not have the same level of success due to the complexity of the models and the high computational costs that prevented a more widespread use. Formant synthesis assumes that the function of vocal organs can be adequately modeled by the simulation of formant frequencies and amplitudes [19]. The strength of formant synthesis lies in its relative simplicity and in the small memory footprint of the engine and voice data. Even though the voice does not sound very natural, all instances of speech sound the same way, which, with some training of the users, leads to an acceptable understanding of the speech.

The Mbrola project uses concatenation synthesis and supports a large number of languages. It can be used freely in non-commercial and non-military applications. This software is not a complete TTS system because the text must first be converted into phonemes and prosodic information. This requires additional software to carry out this transformation prior to invoking the Mbrola software. In the case of the Portuguese language, there is a Perl module called Lingua::PT::Speaker, developed at Universidade do Minho, that deals with this transformation.

Espeak, in turn, is a complete TTS system, as it receives "raw text" as input and produces voice as a result. It is free software and comes bundled in many different Linux distributions. Espeak includes different languages and voices whose characteristics can be modified. It does not use voice synthesis based on concatenations, leaving no significant marks in the generated voice. It uses formant synthesis, which makes the software smaller, allowing several languages to be included in a small package. However, the sound is not as natural as in other systems, since it uses a completely artificial synthesis method.

The RealSpeak TTS engine is based on concatenation algorithms, where actual human voice segments are stored and used to convert any text into speech. In-depth language-specific linguistic knowledge provides intelligent pronunciation of a wide range of variable inputs. The Speech Application Programming Interface (SAPI) is an API developed by Microsoft to provide speech synthesis within Windows applications. The API has been designed to allow the use of speech synthesis through a standard set of interfaces, acces-

sible from a variety of programming languages, allowing the production of third-party TTS applications (it can be viewed as an interface or middleware that sits between applications and speech engines).

The RealSpeak voices from Nuance enterprise are commercial SAPI5 voices that provide high quality speech, available in a variety of languages. The use of these voices with sapi2wav, a free Windows tool for TTS conversion on the command line, provides a convenient interface for high quality speech synthesis. However, our prototype runs on Linux, and it is not possible to port SAPI or sapi2wav due to the unavailability of the source code. The solution was to use Wine, a free software application that allows Unix-like operating systems to execute programs written for Microsoft Windows. This workaround allowed us to use the high quality Madalena European Portuguese Female 22khz voice in our demonstrations.

V. Conclusion

The use of voice provides an additional form of human-machine interaction. Voice interfaces are typically used to improve the user experience, either by providing an additional dimension to existing multimedia systems, or as an alternative form of input/output, whenever more traditional forms of interaction are impractical, unavailable or not usable (e.g., by visually-impaired users). Other possible benefits from the use of voice interfaces include a reduction in operating costs, better security, the creation of new products, greater customer loyalty, the creation/maintenance of an image of modernity and, ultimately, increased revenue.

In this paper, we described a system that provides access to a common service—email—in a non-standard form—a voice-based interface accessible through the telephone network. The main focus of this work was on the usability of the voice interface, namely on a good structure of the voice menus, keeping a balance between low verbosity and ease of use for new users, and the simplicity and consistency of interaction through the numeric keyboard. Other important concerns in the system were security and flexibility; the latter is provided by the numerous customization items available through the web interface and the use of virtual folders that may span different email accounts.

In terms of implementation, the system consists of two modules running in commodity hardware. The first module is a middleware that provides the actual voice interface. It is implemented on a soft PBX, and makes use of existing text-to-speech software and open-source modules for accessing email accounts. The second module is a web-based interface for the configuration of user preferences and email accounts.

Having a complete prototype implementation, the next steps towards the deployment in a production system are to ascertain its usability on a daily basis by a large number of people with different backgrounds and computer-using skills, and to evaluate its scalability to large data sets and distributed collections of email. Further improvement of the system could be made by introducing an adaptive interface which lowers the verbosity as the user's familiarity with the system increases. On a more technical level, further refinement could be made by developing more accurate and finer-grained methods of determining the language, allowing language adjustment to be performed on a paragraph or sentence basis instead of assuming a single language for the entire content of the message. One final aspect in which the system could be improved is the integration of voice commands, which could significantly increase its usability in situations where it is impractical to use a keyboard, e.g., while driving.

References

[1] R.H. Anderson, T.K. Bikson, S.A. Law, and B.M. Mitchell. Universal access to e-mail: Feasibility and societal implications. *Educational Media International*, 34(2):86–87, 1997.

[2] B. Balentine and D.P. Morgan. *How to Build a Speech Recognition Application*. Enterprise Integration Group, 1999.

[3] P. Branco. Challenges for multimodal interfaces towards anyone anywhere accessibility: a position paper. In *WUAUC'01: Proc. 2001 EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing*, pages 26–27. ACM, 2001.

[4] S. Chan, J. Zhang, and X. Fang. A Study of Enterprise User Adoption of VoIP. In *Proc. 4th ISOneWorld Conference and Convention*, 2005.

[5] R. Corkrey and L. Parkinson. Interactive voice response: review of studies 1989–2000. *Behavior Research Methods Instruments and Computers*, 34(3):342–353, 2002.

[6] M. Crispin. Internet message access protocol - version 4rev1. RFC 3501 (Standards Track), March 2003.

[7] B. Duggan and M. Deegan. Considerations in the usage of text to speech (tts) in the creation of natural sounding voice enabled web systems. In *ISICT '03: Proc. 1st Int. Symp. on Information and Communication Technologies*, pages 433–438. Trinity College Dublin, 2003.

[8] M. Edgington. Investigating the limitations of concatenative synthesis. In *Proc. of EuroSpeech'97*, 1997.

[9] D.L. Grover, M.T. King, and C.A. Kushler. Reduced keyboard disambiguating computer, October 6 1998. US Patent 5,818,437.

[10] R. Halstead-Nussloch. The design of phone-based interfaces for consumers. *ACM SIGCHI Bulletin*, 20:347–352, 1989.

[11] J. Hollan, E. Hutchins, and D. Kirsh. Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(2):174–196, 2000.

[12] K. Jones. *View Mail User's Manual*, 1999.

[13] K. Kallinen. Using sounds to present and manage information in computers. In *Proc. of IS-2003*, 2003.

[14] J.L. Klavans and E. Tzoukermann. Machine-readable dictionaries in text-to-speech systems. In *Proc. 15th Intl. Conf. on Computational Linguistics*, pages 971–975. Association for Computational Linguistics, 1994.

[15] F.F.H. Nah, K. Siau, and H. Sheng. The value of mobile applications: a utility company study. *Commun. ACM*, 48(2):85–90, 2005.

[16] T.V. Raman. Emacspeak—a speech interface. In *CHI '96: Proc. SIGCHI Conf. on Human Factors in Computing Systems*, pages 66–71. ACM, 1996.

[17] P. Resnick and R.A. Virzi. Relief from the audio interface blues: expanding the spectrum of menu, list, and form styles. *ACM Trans. Comput.-Hum. Interact.*, 2(2):145–176, 1995.

[18] B. Shneiderman. The limits of speech recognition. *Communications of the ACM*, 43(9):63–65, 2000.

[19] T. Styger and E. Keller. *Fundamentals of speech synthesis and speech recognition: Basic concepts, state of the art, and future challenges*, chapter 6—Formant Synthesis. WileyBlackwell, 1994.

[20] S. Zeadally, F. Siddiqui, and P. Kubher. Voice over ip in intranet and internet environments. *IEE Proceedings-Communications*, 151(3):263–269, 2004.

[21] Asterisk oficial web site. <http://www.asterisk.org/> 2009-08-12.