deBas – A Sinhala Interactive Voice Response System

J.D. Nallathamby, K.K.R. Kariyawasam, H.D. Pullaperuma, D.C. Vithana, S. Jayasena
Department of Computer Science & Engineering,
University of Moratuwa
Moratuwa, Sri Lanka.

Abstract— **Although there are widely used Interactive Voice Response (IVR) systems in many languages today, there is no Sinhala language IVR system yet. This paper talks about an approach taken in developing a complete Sinhala IVR system. It talks about the research carried out in this area, the process taken, the overall design and implementation aspects and the future work that can be carried out in this area. deBas IVR is a complete Sinhala IVR with automatic speech recognition (ASR) and text-to-speech (TTS) synthesis modules that work in compliance with Media Resource Control Protocol (MRCP). In the ASR component, training the acoustic model is done with SphinxTrain, and decoding with PocketSphinx, which are based on Hidden Markov Models (HMM). In the TTS component, AMoRA Sinhala TTS knowledge base is used, which uses Festival speech synthesis engine and a female diphonic voice, built using Festvox voice building tools. Asterisk is used as the IVR gateway and dial-plan interpreter. MRCPv2 protocol has been followed in developing the speech resources, which uses Session Initiation Protocol (SIP) for establishing controlled connections to external media streaming devices and Real-time Transport Protocol (RTP) for media delivery. The language model of the ASR component has been restricted to digits from 0-9 that are commonly used in IVR systems and the set of words used for our demo application. The word-error-rate and the sentence-error-rate of the ASR component are reported to be 31.4% and 54% respectively, as observed in our experiments. In addition to these, we also introduce a new intonation model that can be applied to any existing Sinhala diphonic voices.**

Keywords-Sinhala; Interactive Voice Response; Automatic Speech Recognition; Text-To-Speech;Media Resource Control Protocol; Hidden Markov Model; Word-Error-Rate; Sentence-Error-Rate

I. INTRODUCTION

A. Background

Interactive Voice Response (IVR) is a technology that allows a device or a system (e.g., a computer) to interact with humans using voice. Dual Tone Multi Frequency (DTMF) keypad inputs may be used in addition to or instead of human voice input. For instance, an IVR system can allow customers to interact with a company's database, and service their inquiries on its own. IVR systems can respond with prerecorded or dynamically generated audio to further direct users on how to proceed.

There are many benefits of IVR from a user (customer or citizen) or an organization's perspective, mainly in saving time and money. For example organizations can save money that would otherwise be spent on employing people. IVR applications are found in places like banks where customers call in for credit card inquiries, account balance information queries, reporting frauds and complaints. The air-line industry uses it for ticket reservations, and real-time flight status inquiries. IVR systems can route calls and filter out the important calls that need vital information from trained specialists. The rest of the calls, that need not have special attention, can be solved by the IVR system itself. This benefits not only the organization, but also the customer with a faster service for simpler queries. An IVR system's effectiveness is rated by the percentage of callers who ask to speak to a human operator. The lower the percentage, the more successful the system is.

B. IVR in Sinhala Language

Although there are widely used IVR systems in many languages today, there is no Sinhala language IVR system as yet. Our objective is to build a Sinhala IVR system to satisfy this timely need. In addition to this general need, the Information & Communication Technology Agency (ICTA) of Sri Lanka has a specific requirement for a Sinhala IVR system in the Lanka Gate country portal[1,2] of the Lanka Government Network to provide voice access for citizen services. In this paper we describe the development of our Sinhala IVR system.

C. Related Work

An IVR platform is the server and operating system hardware and the software platforms on which the IVR solutions run. Typically an IVR platform will consist of the following components at minimum. Firstly, there is the call server, which serves as a telephony gateway. It typically may contain hardware such as telephony-gateway-cards, network cards and sound cards. Secondly, there will be a dialogue flow interpreter. Voice XML is W3C's standard XML format for specifying interactive voice dialogues. Thirdly, there are the speech engines.

Asterisk [3] is a well-known open-source IVR platform. Automatic speech recognition (ASR) and text-to-speech (TTS) are important technologies in an IVR system. An ASR unit is used for recognizing user input and a TTS unit is used for producing voice output.

There has been a lot of work in ASR over the years in other languages, but not much work can be found for Sinhala ASR. The Kathana project [3] was a recent attempt to build a Sinhala ASR system. It could recognize digits from 0 to 9 spoken in Sinhala. It uses the Julius ASR engine [4].

On the other hand, TTS systems in many languages have been around for some time, and examples of Sinhala TTS include the Festival-si [5] and AMoRA [6], both of which are based on the Festival speech synthesis engine [7]. The AMoRA TTS system however is more advanced than the Festival-si TTS system, since it incorporates a phrase breaking model that improves the naturalness of the voice. However both voices lack variation in pitch and pace which contributes towards reducing the naturalness of the voice.

We have considered using Kathana and AMoRA as potential ASR and TTS engines in our IVR system, and also to look into improving these approaches.

In integrating the speech engines to an IVR platform, there are two popular methods: Media Resource Control Protocol (MRCP) [8] and Application Programming Interface (API). MRCP is a protocol that allows interoperability between multiple media resources. It allows diverse elements such as speech engines, text to speech engines, signal generators, signal detectors, fax resources, platforms, and PBX systems to communicate. The resource that is providing the service is known as the MRCP Server and the resource that is requesting the service is known as the MRCP Client [9].

There are two versions of MRCP: version 1 and version 2. Version 1 uses the Real-Time Streaming Protocol (RTSP) and version 2 uses the Session Initiated Protocol (SIP) to help establish control connections to external media streaming devices. In addition, MRCP uses media delivery mechanisms like Real Time Protocol (RTP) [9].

An API is a vendor and application specific method of communication with a resource. The creator of the resource would publish a set of methods which the service requesting application could use to directly invoke the service [9].

II. METHODOLOGY

A. *Introduction*

As stated above, either the MRCP approach or the API approach can be used to integrate speech engines to an IVR system.

When choosing between the two approaches, there are few tradeoffs to consider. A major benefit of MRCP would be one consistent interface for all speech components. If we happen to change the speech server or client the other components need not to be changed, improving portability. Also MRCP uses a standard method of communicating across networks. Therefore the IVR system can be distributed on different physical hardware.

On the other hand, the API approach has its advantages as well. Firstly, we could have more functionality than when following the MRCP approach, in which we are restricting ourselves only to the functionality agreed upon; any additional functionality provided by a specific speech vendor for example, cannot be used. Secondly, adding and removing functionality will also be easy and quick. Thirdly, generally standards do not tend to care much about backward compatibility; for example, MRCPv1 and MRCPv2 are completely different protocols, not compatible with each other. With an API, we could ensure not to break things when vendors change or upgrade. Fourthly, with MRCP, it can take a considerable amount of work and time to get one up and running from scratch [8].

B. *Our Approach and the Components*

The above factors were considered and we selected the MRCP method as our approach. In deciding which method of integration was best for us, the MRCP method or API method, we had to factor out all the pros and cons of the two methods and went with MRCP. The overall architecture of our system is shown in figure 1.

One main requirement here was 'flexibility' which we would have if we follow a protocol. We need to be able to switch from one speech system provider/vendor to another, with the least switching time. In addition, we found that building an MRCP-compliant speech server wasn't that hard as we were able to find an open source cross platform project called UniMRCP [10]. It provides many things required for the implementation and deployment of both an MRCP client and an MRCP server as well as the architecture to plug in any speech server of our choice, by implementing a set of methods (See figure 2).

Our project involves number of components that are distributed and communicate with each other (Fig. 1). Early in the project, having identified these components, we searched for existing solutions that implement these components, fully or partly, and evaluated them for suitability and identified those that need further development. Some had multiple versions of products, while some were reputed as the best. Table 1 shows the final list of components we decided to have for our IVR system and the products we chose to use as the basis for the components.

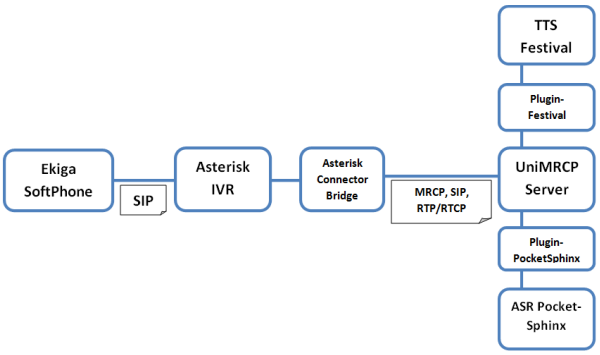

Figure 1: MRCP-based Overall System Architecture of IVR System

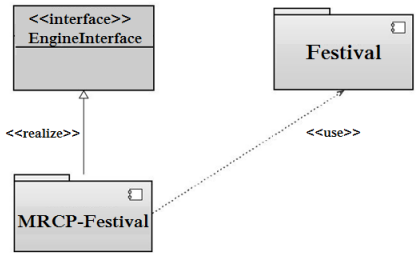

Figure 2: UniMRCP Plug-in Architecture

Table 1: Main Components in our IVR System

Component	Software Product
ASR engine	*Pocketsphinx*
TTS engine	*Festival* (*AMoRA*)
IVR platform	*Asterisk*

We chose Pocketsphinx [11] as our ASR engine because there is a plug-in for UniMRCP, with an implemented basic set of MRCP requests, responses and events. Our initial tests with the engine were very promising and Section 3 shows the word error rates (WER) in our tests. Also Pocketsphinx, like UniMRCP, is written in C language, which makes it much smaller, faster and portable compared to other large ASR engines.

The choice of the TTS engine was straight forward since the AMoRA project had already used the Festival speech synthesis engine. Here we had to implement a new plug-in for Festival to connect with UniMRCP.

For our IVR platform, Asterisk was a straightforward choice as there was already a UniMRCP client by the name of 'Asterisk connector bridge' [12]. Also the extensive features in Asterisk made it an obvious choice. Finally, as we needed an open source SIP phone for our testing environment, we selected Ekiga [13]. Our development and deployment were carried out completely in a Linux environment, and it ensured compatibility among all the components.

C. *Connecting the Components*

The first step of our implementation was to connect the available components. The Asterisk connector bridge provides two methods, MRCPRecog() for recognition and MRCPSynth() for synthesis. Both methods take as arguments a mandatory field and some optional fields. The mandatory field for MRCPRecog() would be the ASR grammar in Java Speech Grammar Format (JSGF) [14]. The mandatory field for the MRCPSynth() function would be the text to be converted to speech. The optional parameters specify the header content of the MRCP message [9]. These functions can be directly invoked from the dial-plan of Asterisk. In order for the communication to work we have to make sure the IP/ports for receiving and transmitting data are correctly configured for each protocol. Between the Asterisk client and the UniMRCP server, we need to configure ports for SIP (since we decided to use MRCPv2) and RTP. We also needed to specify, the ASR grammar format, and TTS text format in the UniMRCP configuration.

In order to connect a call from Ekiga to Asterisk we needed to create a SIP account in Ekiga and explicitly give permission to the particular node to connect with Asterisk. In addition, to be able to run Ekiga and Asterisk on the same local machine, we changed the listening SIP port of Ekiga to another port, because by default Ekiga and Asterisk will be listening on the same port, 5060 [15].

Next we wrote dial plans in Asterisk using MRCPRecog() and MRCPSynth() in addition to the standard set of dial plan applications provided by Asterisk[16]. Another task we accomplished was parsing the recognition result from data carried in the MRCP message body of the RECOGNITION-COMPLETE event in Natural Language Semantics Markup Language (NLSML) format [9]. Asterisk does not provide a default method to parse the result. Therefore using the Asterisk Gateway Interface (AGI) we invoke a PERL script upon receiving the result and parse the NLSML format and get the recognized grammar from it [17].

Another improvement we did on the UniMRCP server was to implement the speech-complete-timeout (SCT) specified in the MRCP protocol, but not implemented in the present version of UniMRCP [9]. What this means is that UniMRCP no longer stops recognition as soon as it detects voice inactivity, but continues recognition until the SCT completes, thus we are now able to detect multiple words. This parameter can be specified in the MRCPRecog() method; if not, default value will be used.

D. *Sinhala Automatic Speech Recognition (ASR)*

According to the literature on ASR there are 3 models that are necessary to do speech recognition; an acoustic model (AM), a language model (LM) and a phonetic dictionary (Dic). The AM will be created by taking audio recordings of speech and their transcriptions, and 'compiling' them into statistical representations of the sounds that make up each word ('training'). The LM will contain the probabilities of sequence of words. The LM is necessary for compiling the AM. The Dic contains a mapping from words to phones. The representation of the words and their corresponding phones are important data for accurate decoding. During the decoding stage the LM may be replaced by a grammar. A grammar is a much smaller file containing sets of predefined combinations of words. Grammars are most useful in IVR type applications. The training for our model was done using SphinxTrain; a tool provided by CMUSphinx for training AMs. SphinxTrain creates a statistical model for the data prepared based on Hidden Markov Models (HMM). The LM is generated by a web based tool known as LMTool; again provided by CMUSphinx for building uncomplicated small LMs. In addition, the LMTool also generates a Dic, which we may use or replace with our own Dic.

An important step during the training process is the creation of audio files. Audacity [18] helps to see the waveform we record and thus can clearly identify the

presence of background noise and its amplitude relative to our voice.

In training the AM we needed to do multiple rounds of training to get the best results. It is critical to test the quality of the trained database in order to select best parameters and understand how the IVR application performs and optimizes performance. To do that, a test decoding step is needed. This step is also provided by CMUSphinx. The test gives us a word error rate (WER) and a sentence error rate (SER).

$$WER = (S + 0.5D + 0.5I)/N$$

Where,

S is the number of substitutions,
D is the number of the deletions,
I is the number of the insertions,
N is the number of words in the reference.

Our first model to train PocketSphinx was trained with 400 voice prompts which are taken from 4 different persons 100 each. For the LM we used the word representation and phoneme representation used by Kathana. We carried out tests by reducing the number of training prompts (100X3, 100X2, 100X1). We were expecting the WER for models with lesser trainers to be high but apparently it was vice-versa.

Then we started changing the LM and repeated the tests. This time for the same word representation of Kathana we used the phonemes produced by LMTool. Next we tried a word representation which we got from referring to the Sri Lanka Standards (SLS) [18]. The problem with the phoneme representations generated by the LMTool is that they are based on the English pronunciations of the Sinhala words. So therefore we used the SLS word representation and derived a phone set for the Sinhala words in our dictionary using the 39 standard phonemes defined by CMUSphinx in the CMU Pronouncing Dictionary [19]. Each step gave us declining WER values.

Finally we generated an AM trained with 1400 (4 persons, 350 each) prompts. We tested the models with 50 test prompts taken from one of the 4 persons who trained the model. The test results we got for each change we did are tabulated in section 3.

E. Sinhala Text-To-Speech (TTS) Synthesis

To add Sinhala TTS functionality to our IVR, we needed to plug in Festival engine to UniMRCP. We used the BSD socket interface provided by Festival, where the Festival engine works as the server, and we wrote the client program to access it. Basically the server offers a new command interpreter for each client that attaches to it.

Festival phrase break CART (Classification And Regression Tree) uses categorized word bases to predict phrase breaks in a given input text [20]. For the Sinhala language, AMoRA has used function words, punctuation marks, adjectives, pre verbs, and participles etc. as part-of-speech (POS) tags for allocating phrase breaks. We have expanded the POS tags; thereby increasing the span of the phrase breaking process.

The common problem in any TTS system is its artificialness of the synthesized voice; however the techniques of phrasing model, intonation model and duration model can help to bring it closer to the natural voice.

We have introduced a new intonation model. Intonation is generally of two types, accent positioning and F0 generation [21]. Here we have developed the model from F0 generation.

We have considered 400 Sinhala sentences in developing the intonation model. The steps for creating an intonation model are as follows:

- Designing of database
- Synthesizing prompts (for labeling)
- Recording of prompts
- Phonetically labeling of prompts
- Extract pitch marks and F0 contour
- Build utterance structures
- Extract features for prediction & build feature description files
- Build regression model to predict F0 at start, middle and end of syllable
- Construct scheme file with F0 model that can be incorporated to AMoRA system

III. RESULTS AND DISCUSSION

Table 2: Recognition Accuracy Rates of Pocketsphinx

Model (word representation - phonetic representation)	SER	WER
4 person 100X4 (Kathana)	92.0%	77.4%
3 person 100X3 (Kathana)	80.0%	58.4%
2 person 100X2 (Kathana)	80.0%	56.2%
1 person different user (Kathana)	90.0%	94.9%
1 person same user (Kathana)	74.0%	45.3%
4 person(100x4) (Kathana - LMTool)	90.0%	67.9%
4 person (100x4) (SLS - LMTool)	86.0%	56.9%
4 person 350X4(SLS-LMTool)	66.0%	31.4%
3 person 350X3(SLS-LMTool)	68.0%	35.8%
2 person 350x2 (SLS-LMTool)	60%	32.1%
1 person different user (SLS-LMTool)	94.0%	92.0%
1 person same user (SLS-LMTool)	74.0%	56.2%
4 person 350X4 (SLS-CMU Dic)	54.0%	31.4%

From the above results we conclude that the best WER is obtained for the acoustic model trained by members in our group with a set of 350 prompts each.

The TTS part of the project is an improvement of the existing project; the best evaluation method would be comparing the improved system with existing system. Therefore we have used Mean Opinion Score (MOS) method to evaluate naturalness of the generated new voice compared to the existing voice.

MOS is a subjective test. MOS test has been used in telephony networks to identify human user views regarding the network [22]. MOS test provides a numerical indication about the system. This test uses five integers (1-5), where 1 is represent lowest quality and 5 is represent best quality. The grading criteria given to listeners are shown in table 3. This test was carried out for existing TTS systems (AMORA-TTS and UCSC-TTS) and the improved TTS system. The new improved TTS consists of 2 intonation models that we created by selecting a subset each of 250 sentences from the set of 400 sentences. Model A consists of sentences with question marks and full stops, whereas model B consists of only sentences with question marks.

The UCSC voice lacks a phrasing model, so to make a fair comparison we incorporated the phrasing model of project AMoRA to UCSC voice as well.

Table 3: Mean Opinion Score Test (MOS)

MOS	Quality
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

Following criteria was followed in collecting the test data.

- 50 listeners were selected. All listeners were native speakers of Sinhala.
- Each listener was unaware as to which system they are listening.
- Each listener listen to 10 sentences and they were asked to select the MOS value.
- 10 sentences were selected on random form 400 sentences.

Based on the test we got the results shown in figure 3.

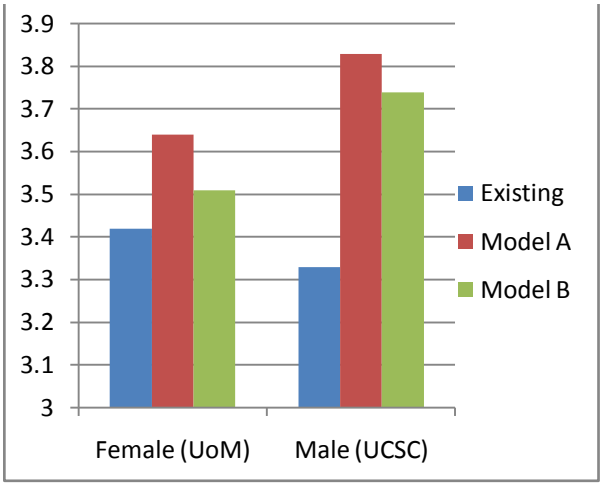

Figure 3: UniMRCP Plug-in Architecture

According to the MOS test results we found that the male voice was much better than the female voice with the new intonation model, specifically with model A.

IV. Conclusion And Future Works

At this point of time the objectives of deBas have been realized and deBas can be used as a practical IVR system in the Sinhala language and as an area of research in general IVR. To the best of our knowledge this is the first ever Sinhala IVR research product to be developed following the MRCP protocol. It comes completely with a basic set of functions that can be used to deploy any scale of IVR applications in Sinhala language. The effective word-error-rate and sentence-error-rate of the speech recognition module is observed as 31.4% and 54% respectively.

There are some areas open for research in deBas. Future research and development can be based on three areas which are the MRCP protocol, PocketSphinx ASR and Festival TTS.

Some of the request header parameters of the MRCP protocol have not yet been implemented in our solution. E.g. in the RECOGNIZE method header of the ASR module, we find parameters such as confidence threshold, speech-incomplete-timeout, speed-vs-accuracy etc. Certainly these can be implemented, but since they depend on the ASR engine and the TTS engine, you cannot develop those in isolation without considering the features of the ASR and the TTS module. The ASR and TTS module limitations will also have an effect in this development.

The PocketSphinx ASR is still in its early stages. There is still active development being carried out, and as and when features are added, deBas can be improved by incorporating those features. In addition, work can also be carried out in areas to improve accuracy levels of the system. Noise cancellation techniques are widely studied today. These techniques can be incorporated to the PocketSphinx ASR module. Another area for research is the development of a phonetic dictionary for Sinhala.

In the TTS module also, there is future work that can be carried out. Development of accent and boundary tone module, implementing domain specific F0 and duration models for multiple domains, enhancement of phrase breaking algorithm by introducing more phrasing rules and implementation of intonation and duration models using other approaches mentioned in the literature survey (e.g. Intonation models using ToBI etc.) are some of the areas where significant work can be carried out.

REFERENCES

[1] Sri Lanka Country Portal [Online]. Available: http://www.srilanka.lk [Accessed: 15-Jun-2011]

[2] Lanka Gate Developer Portal [Online]. Available: http://developer.icta.lk/ [Accessed: 15-Jun-2011]

[3] Kathana Sinhala Speech Recognition [Online]. Available: http://kathana.net23.net/index.html [Accessed: 15-Jun-2011]

[4] Open-Source Large Vocabulary CSR Engine Julius [Online]. Available: http://julius.sourceforge.jp/en_index.php [Accessed: 15-Jun-2011]

[5] Festival-si [Online]. Available: http://www.ucsc.cmb.ac.lk/ltrl/?page=panl10n_p1&lang=en#tts [Accessed: 15-Jun-2011]

[6] New Prosodic Phrasing Model for Sinhala Language, W.M.C. Bandara, W.M.S. Lakmal, T.D. Liyanagama and S.V. Bulathsinghala, Prof. Gihan Dias, Dr. Sanath Jayasena

[7] The Festival Speech Synthesis System [Online]. Available: http://www.cstr.ed.ac.uk/projects/festival/ [Accessed: 15-Jun-2011]

[8] API or MRCP Integration: Choosing a Development Path for Speech Recognition Solutions [Online]. Available: http://www.lumenvox.com/resources/whitepapers/apiMRCPIntegration.aspx [Accessed: 15-Jun-2011]

[9] A Media Resource Control Protocol (MRCP) Developed by Cisco, Nuance, and Speechworks [Online]. Available: http://www.ietf.org/rfc/rfc4463.txt [Accessed: 15-Jun-2011]

[10] UniMRCP: Open Source MRCP Project [Online]. Available: http://www.unimrcp.org/ [Accessed: 15-Jun-2011]

[11] Pocketsphinx: Automatic Speech Recognition Engine [Online]. Available: http://cmusphinx.sourceforge.net/2010/03/pocketsphinx-0-6-release/ [Accessed: 15-Jun-2011]

[12] Asterisk connector bridge [Online]. Available: http://code.google.com/p/unimrcp/wiki/asteriskUniMRCP [Accessed: 15-Jun-2011]

[13] Ekiga softphone [Online]. Available: http://ekiga.org/ [Accessed: 15-Jun-2011]

[14] Java Speech Grammar Format [Online]. Available: http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/ [Accessed: 15-Jun-2011]

[15] Ekiga as an Asterisk client [Online]. Available: http://wiki.ekiga.org/index.php/Ekiga_as_an_Asterisk_client [Accessed: 15-Jun-2011]

[16] Asterisk Dialplan Applications [Online]. Available: https://wiki.asterisk.org/wiki/display/AST/Dialplan+Applications [Accessed: 15-Jun-2011]

[17] [17]AGI Commands [Online]. Available: https://wiki.asterisk.org/wiki/display/AST/AGI+Commands [Accessed: 15-Jun-2011]

[18] Sri Lanka Standard Sinhala Character Code for Information Interchange, SLS 1134 : 2004 [Online]. Available: http://www.locallanguages.lk/files/Sri_Lanka_Standard_Sinhala_Character_Code_for_Information_Interchange_SLS_1134_-_2004.pdf [Accessed: 15-Jun-2011]

[19] CMU Pronouncing Dictionary [Online]. Available: http://www.speech.cs.cmu.edu/cgi-bin/cmudict#phones [Accessed: 15-Jun-2011]

[20] Building Prosodic Models [Online]. Available: http://festvox.org/bsv/c1637.html#AEN1639 [Accessed: 15-Jun-2011]

[21] F0 Generation [Online]. Available: http://festvox.org/bsv/x1803.html [Accessed: 15-Jun-2011]

[22] Assessing Text-to-Speech System Quality, White Paper by SpeechWorks Solutions Division, ScanSoft Inc, 2004.