TECHNICAL REPORT RDMR-WD-10-40

TEXT TO SPEECH (TTS) CAPABILITIES FOR THE COMMON DRIVER TRAINER (CDT)

Michael Hanners
And
Danny Carter
Weapons Development and Integration Directorate
Aviation and Missile Research, Development, and Engineering Center

October 2010

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 074-0188

1. AGENCY USE ONLY	2. REPORT DATE October 2010	3. REPORT TYPE AND DATES COVERED Final

4. TITLE AND SUBTITLE Text To Speech (TTS) Capabilities for the Common Driver Trainer (CDT)	5. FUNDING NUMBERS
6. AUTHOR(S) Michael Hanners and Danny Carter	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Commander, U.S. Army Research, Development, and Engineering Command ATTN: RDMR-WDG-C Redstone Arsenal, AL 35898	8. PERFORMING ORGANIZATION REPORT NUMBER TR-RDMR-WD-10-40
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE A

13. ABSTRACT *(Maximum 200 Words)*

The U.S. Army Aviation and Missile Research, Development, and Engineering Center (AMRDEC) was tasked to analyze options for potential Text To Speech (TTS) improvements as part of the scenario generation system for the Common Driver Trainer (CDT). This report provides a discussion of some of the available TTS technologies, as well as a recommended path forward. In addition to the research performed, AMRDEC personnel also developed code to assist in familiarization with TTS capability. Descriptions of this software, along with code samples from some of the commercial products, are provided in the report. AMRDEC recommended that the CDT program develop an application or component that provides TTS capabilities, including text and file input, voice selection, a custom dictionary, voice recording, file exporting to standard audio formats, and sound file organization.

14. SUBJECT TERMS Text To Speech (TTS), audio, training device, trainers, Common Driver Trainer (CDT), software	15. NUMBER OF PAGES 33
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

EXECUTIVE SUMMARY

The Common Driver Trainer (CDT) program is in the process of upgrading software, including replacement of the scenario-generation capability. The U.S. Army Aviation and Missile Research, Development, and Engineering Center (AMRDEC) is providing support to the CDT program and was tasked to analyze options for potential Text to Speech (TTS) improvement as part of the scenario generation system. This report provides a discussion of some of the available TTS technologies, as well as a recommended path forward. A decision matrix was developed that summarizes the findings for the technologies evaluated, resulting in a commercial product by Loquendo being selected as providing the best functionality. In addition to the research performed, AMRDEC personnel also developed code to assist in familiarization with TTS capability. Descriptions of this software, along with code samples from some of the commercial products, are provided in the report.

Regardless of the TTS technology selected, AMRDEC recommends that the CDT program develop an application or component that provides TTS capabilities, including text and file input, voice selection, a custom dictionary, voice recording, file exporting to standard audio formats, and sound file organization. This software component can be developed independently from other scenario-generation system development efforts and may either be kept as a separate tool or integrated with other CDT software as required.

TABLE OF CONTENTS

Page

I. INTRODUCTION 1

II. CDT OVERVIEW 1

III. SPECIFIC CDT REQUIREMENTS 2

IV. COMMERCIAL OPTIONS 3

A. Microsoft 3
B. Wizzard Software 4
C. NeoSpeech 8
D. Cepstral 9
E. NaturalReader 10
F. Digital Future 11
G. Loquendo 13

V. SAMPLE APPLICATIONS USING MICROSOFT LIBRARIES 17

A. C++ Application 17
B. C# Application 19

VI. RECOMMENDATION 20

REFERENCES 23

LIST OF ACRONYMS AND ABBREVIATIONS 24

LIST OF ILLUSTRATIONS

Figure	Title	Page
1.	Common Driver Trainer Stryker Variant [1]	2
2.	Creating an AT&T Natural Voices TTS Engine [5]	4
3.	Initialization and Shut Down of the Wizzard TTS Engine [5]	5
4.	Speaking Text Using the Wizzard TTS Engine [5]	5
5.	Rate and Volume Adjustment for Wizzard SDK [5]	5
6.	Speech Synthesis Markup Language Tags [5]	6
7.	AT&T Natural Voices Dictionary Editor [4]	7
8.	Various Code Samples from the VoiceText SDK [7]	8
9.	Cepstral's Pitch Control	9
10.	Cepstral's Emphasis Control	9
11.	NaturalReader Examples [9]	10
12.	NaturalReader Pronunciation Editor [9]	11
13.	TextSpeech Pro Application	11
14.	TextSpeech Pro Pronunciation Corrector	12
15.	Various Code Samples from the VoiceText SDK [9]	13
16.	Code Sample Showing Use of the Loquendo SDK [11]	14
17.	TTS Director [12]	15
18.	TTS Director Unit Selector [12]	16
19.	Lexicon Manager Tool [12]	17
20.	C++ TTS Application	18
21.	C++ SAPI Code	18
22.	C# TTS Application	19
23.	Scope Statements in C# TTS Application	19

LIST OF ILLUSTRATIONS (CONCLUDED)

Figure	Title	Page
24.	Initializing the Engine	19
25.	Adding Installed Voices to a Combo Box	20
26.	Speaking Text	20

LIST OF TABLES

Table	Title	Page
1.	Wizzard Software AT&T Natural Voices System Requirements [4]	7
2.	NeoSpeech VoiceText System Requirements [6]	8
3.	Cepstral TTS System Requirements [8]	9
4.	NaturalReader System Requirements [9]	10
5.	Programming Languages Supported	12
6.	Loquendo System Requirements [11]	13
7.	Decision Matrix	21

I. INTRODUCTION

The Common Driver Trainer (CDT) program is in the process of revamping its scenario generation capability. One feature of the scenario-generation tool is the ability to add sounds to occur at various points in the scenario. In support of this capability, there is a requirement to provide Text to Speech (TTS) capability, such that verbal cues can be provided as desired. As part of this new development, the Aviation and Missile Research, Development, and Engineering Center (AMRDEC) was asked to research TTS options in support of the CDT program. This report includes reviews of various commercial options of various tools that provide speech synthesis capability.

Speech synthesis is artificial production of sound by a computer that resembles human voices. This technology gives computer programmers the ability to match human-like voices to computer-generated simulations. TTS is a tool that uses speech synthesis to allow programmers to provide input text for computer-generated speech synthesized voices to speak. Some voices are very realistic, while others are somewhat robotic. This study describes features from some of the available technologies, along with a description of software developed by AMRDEC. Finally, a recommendation is provided to the CDT program as to a potential path forward.

II. CDT OVERVIEW

The CDT provides both initial and sustainment training for several families of vehicles, including the Stryker (Fig. 1), Abrams, and Mine Resistant Ambush Protected (MRAP) vehicles. The components of the CDT include the vehicle cab, the visual system, the Instructor/Operator Station (IOS), the After Action Review (AAR) station, the motion system, and various computers for the simulation and image generation [1].

The vehicle cab is interchangeable between the various platforms supported by the CDT. This ensures commonality among many of the components for the various vehicle types supported. The display system is controlled by a commercial Image Generator (IG), specifically the Rockwell Collins EPX-50. The host simulation computer communicates to the EPX IG using the Common Image Generator Interface (CIGI) standard.

The IOS allows the instructor to select the scenario for the student to execute and to monitor the student's progress during the exercise. Additionally, the instructor can provide verbal instructions to the student through the intercom system and insert system faults. Finally, the IOS supports recorded AAR for reviewing the student's performance.

When generating scenarios, the instructor must have the ability to insert audio files for playback during an exercise. This can be accomplished through both insertion of pre-recorded audio files and using TTS to generate new audio files.

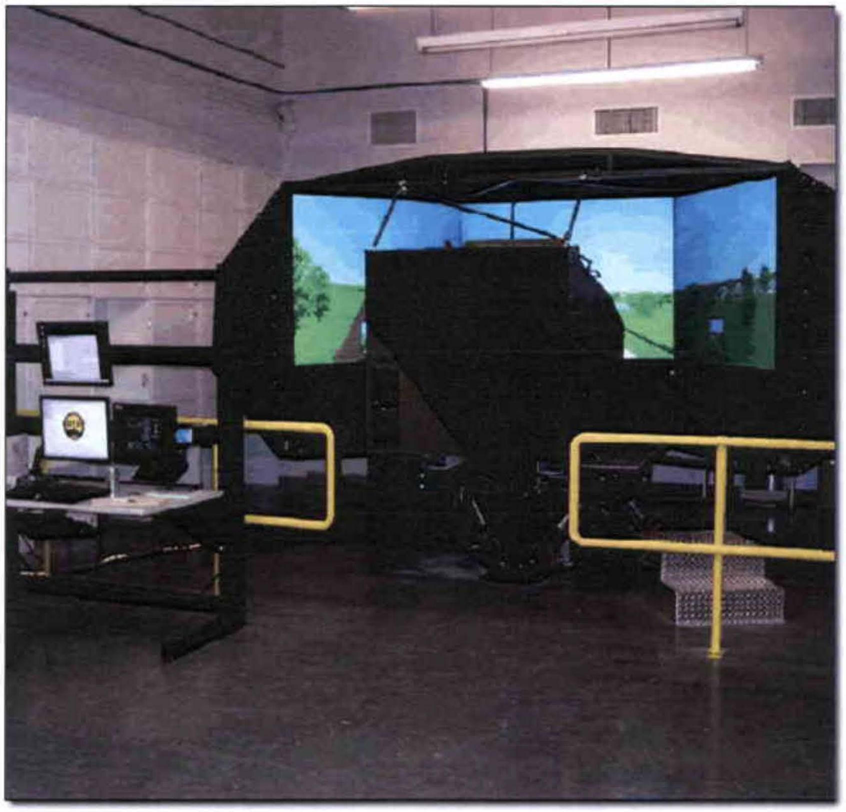


Figure 1. Common Driver Trainer Stryker Variant [1]

III. SPECIFIC CDT REQUIREMENTS

The CDT requirements related to TTS and sound recording are as follows [2]:

- The CDT scenario-generation system shall allow a Scenario Manager to add new sound libraries.
- The CDT scenario-generation system shall allow a Scenario Manager to add new sound files to existing sound libraries.
- The CDT scenario-generation system shall include a recording/editing capability to facilitate the creation of new aural cues.
- The recording system shall be capable of recording human speech.
- The recording system shall be capable of editing and saving sound files in the following formats: .wav, .mp3, .wma.
- The recording system shall allow the Scenario Manager to set default sound file parameters, such as sampling rate, for each supported file format.
- The CDT scenario-generation system shall include a TTS speech synthesis capability to facilitate the creation of new aural cues from text files.
- The CDT scenario-generation system shall allow a Scenario Author to designate whether a sound file associated with a scripted event is to be played through the cab speakers, the cab headset, or both.
- The CDT scenario-generation system shall be capable of storing 1024 sound files or calls to sound files as part of a scenario file.

IV. COMMERCIAL OPTIONS

There are many choices for TTS software vendors available, most of which offer the same basic options. Some of the vendors actually use the same voices. The following sections describe some of the various commercial options available.

A. Microsoft

Microsoft has developed a Speech Application Programming Interface (SAPI) that allows the use of speech synthesis and speech recognition within Windows applications. All versions of the Application Programming Interfaces (APIs) are designed to allow software developers to incorporate into programs speech synthesis and speech recognition by using a standard set of interfaces. These APIs are accessible by a number of programming languages.

In earlier versions, the SAPIs were used to act as an interface between applications and the speech engine. In 2000, Microsoft released SAPI 5 in which applications communicate with a runtime component called the sapi.dll. Applications make calls to the engine through the API, and the runtime component interprets and processes the commands. Section V.A. shows an example of a simple application that was developed by AMRDEC using the Microsoft SAPI library.

SAPI 5.0 is a complete redesign from earlier versions. Significant alterations were needed to make older engines and applications compatible with this new version. This new change was to prevent the API from relying on a particular engine. Also, the change was made to make it easier to incorporate speech in applications by moving initialization code into the runtime.

SAPI 5.1 was released in 2001 as part of the Speech Software Developer Kit (SDK) 5.1. Interfaces were added to the API to allow use by Visual Basic, scripting languages, and managed code. This version was shipped in Windows XP.

SAPI 5.2 was a special version of the API that was released in 2004 that is for use only in the Microsoft Speech Server. It added support for Speech Synthesis Markup Language (SSML) that allows changes for special pronunciation of words such as acronyms. Also added in the new version was Speech Recognition Grammar Specification (SRGS). This capability added the ability for speech recognition that might be used in automated phone services asking for a certain phone extension.

SAPI 5.3 was the version of the API that came with Windows Vista. This version provides new recognition and speech engines. With Windows Speech Recognition being part of the operating system, the Speech SDK and APIs are now part of the Windows SDK. SAPI 5.3 supports W3C XML speech grammars for recognition and synthesis. The SSML version 1.0 adds ability to mark up voice characteristics, speed, volume, pitch, and pronunciation.

A managed code Speech API with similar functionality to SAPI 5 is standard in the .NET Framework 3.0. This API is available in Windows XP, Windows Server 2003/2008, and Windows Vista [3]. Section V.B. shows example code developed by AMRDEC using the Microsoft managed code Speech API.

B. Wizzard Software

Wizzard Software offers packages that allow developers to integrate speech into their projects using either AT&T Natural Voices Desktop or IBM ViaVoice. The IBM ViaVoice technology provides a nearly unlimited quantity of voices, since the voices are digitized by modifying various parameters such as pitch and gender at run-time. However, the AT&T Natural Voices technology likely provides the best fit for the CDT, since the voices that are supported have a more natural sound [4].

Wizzard Software offers a cross platform development suite that allows for both Windows and Linux users. The AT&T Natural Voices engine provides a means to customize pronunciation of words, along with defining acronyms and abbreviations.

Wizzard Software's SDK can be purchased for $295. However, licensing is also required to use, deploy, or distribute speech technologies or audio files. Desktop deployment is $10.50 per copy with a minimum order of $1,500.

To create an engine for the Windows Desktop Edition of the AT&T Natural Voices TTS Engine, the code shown in Figure 2 is required.

```
CTTSEngine *pEngine = 0;
TTSConfig ttsConfig;
// Setup our configuration
ttsConfig.m_eEngineModel = TTSENGINEMODEL_STANDALONE;
ttsConfig.m_eEngineBehavior = TTSENGINE_SYNCHRONOUS;
// Create the engine
result = ttsCreateEngine(&pEngine, ttsConfig);
// Success?
if (result == TTS_OK && this->m_pEngine) {
      // AddRef() the engine
      pEngine->AddRef();
      // application continues
            . . . .
      // Release the engine
      pEngine->Release();
}
```

Figure 2. Creating an AT&T Natural Voices TTS Engine [5]

After the engine is created, it must be initialized. Once this is completed, standard TTS actions can occur. When these functions are finished, the engine should be shut down. So, for every call to initialize, there should be a call to shut down. This is demonstrated in Figure 3.

```
// Initialize the engine
TTS_RESULT result = pEngine->Initialize();
if (SUCCEEDED(result)) {
      . . . speak etc.
      // we are done, so shut down the engine
      pEngine->Shutdown();
}
```

Figure 3. Initialization and Shut Down of the Wizzard TTS Engine [5]

After the engine is initialized, the voices can be enumerated and a specific voice for the application can be selected. The code for this may also be found in the Developer's Guide. Prompting the engine to speak the requested text is performed as shown in Figure 4.

```
// Speak some text
string szText = "hello!";
TTS_RESULT result = pEngine->Speak((PUTF8String) szText.c_str(),
CTTSEngine::sf_default);
if (FAILED(result)) {
      cout << CTTSResult::GetErrorString(result);
}
```

Figure 4. Speaking Text Using the Wizzard TTS Engine [5]

Additional settings can be made, such as rate and volume, as shown in Figure 5.

```
// Rate and volume adjustment

pEngine->SetRate(-10); // Approximately 1/3 normal rate
pEngine->SetVolume(25); // ¼ normal speaking volume
pEngine->Speak((PCUTF8String)''A slow whisper'', 14,
      CTTSEngine::sf_default);
pEngine->SetRate(10); // Approximately 3 times normal rate
pEngine->SetVolume(200); // twice normal speaking volume
pEngine->Speak((PCUTF8String)''A quick shout!'', 14,
      CTTSEngine::sf_default);
```

Figure 5. Rate and Volume Adjustment for Wizzard SDK [5]

SSML provides tags that allow a programmer to control various aspects of speech synthesis such as pronunciation, rate, and pitch. Figure 6 shows examples of "Say-As" tags which are used to manipulate how the words are to be spoken.

Say-As
Say-As tags provide contextual hints to the TTS engine about how text should be pronounced. The TTS engine supports a number of different contexts that can be used to fine-tune the pronunciation of words.

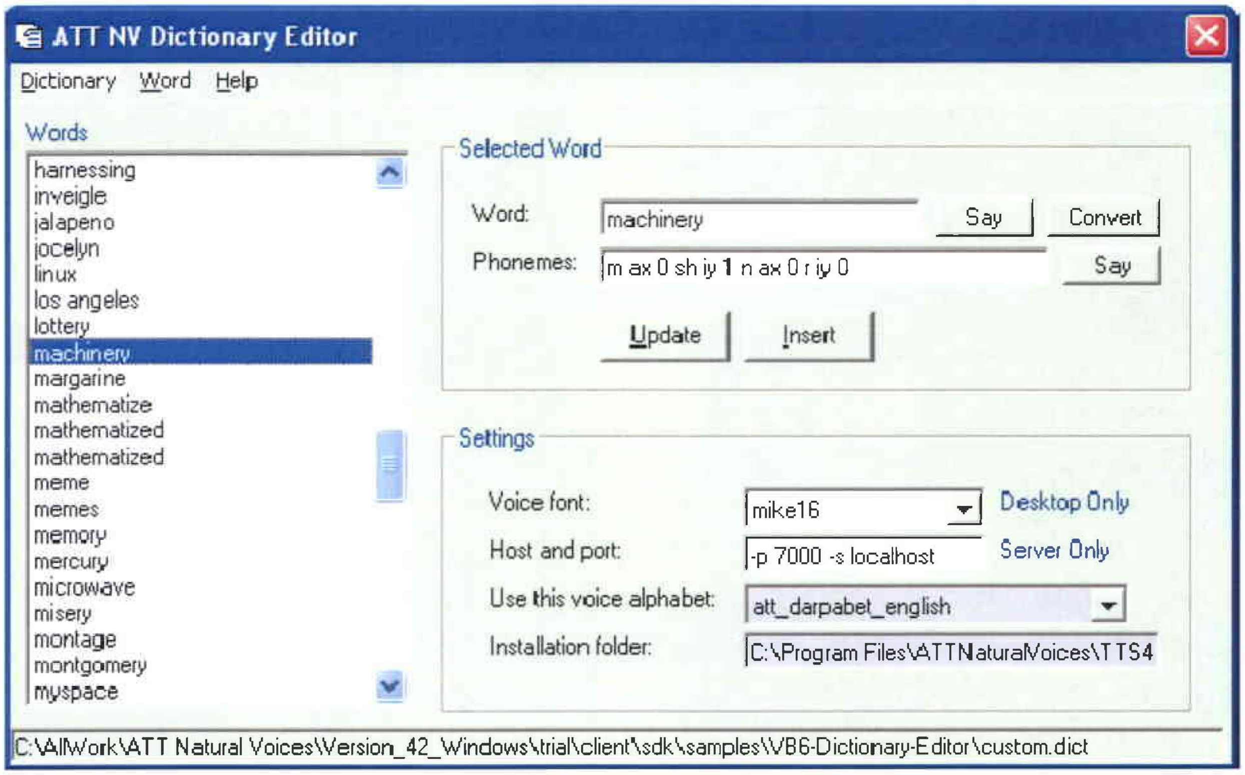
Say-As > Acronym
The Acronym context tells the TTS engine to treat the text as an acronym and to pronounce the text as the letters in the words. This tag is especially useful if your text is mostly upper case and you use the ATT_Ignore_Case tag but then encounter an acronym. Syntax:
<SAY-AS Type="Acronym"> text </SAY-AS>
Example: MADD <Say-as type="acronym"> MADD </Say-as>
Note: Pronounced as "mad M-A-D-D".

Say-As > Address
The Address context tells the TTS engine to treat the text as an address. Syntax: <SAY-AS Type="Address"> text </SAY-AS>
Example: <Say-as Type="Address"> 123 Main St. , New York, NY 10017 </Say-as>
Note: Will be pronounced "one twenty three main street, New York, New York one zero zero one seven"

Say-As > ATT_Math
The ATT_Math context tells the TTS engine to treat the text as a mathematical expression.
Syntax: <SAY-AS Type="ATT_Math"> text </SAY-AS>
Example: <Say-as Type="ATT_Math"> 3+4=7 </Say-as> 3+5=8
Note: Pronounced as "three plus four equals seven three plus sign five equal sign eight"

Figure 6. SSML Tags [5]

The Wizzard Software AT&T Natural Voices SDK also comes with a Dictionary Editor, which can be used as delivered or used as a baseline for a custom application. This application, shown in Figure 7, allows users to provide custom pronunciations to words.



Words: harnessing, inveigle, jalapeno, jocelyn, linux, los angeles, lottery, machinery, margarine, mathematize, mathematized, mathematized, meme, memes, memory, mercury, microwave, misery, montage, montgomery, myspace

Word: machinery

Phonemes: m ax 0 sh iy 1 n ax 0 r iy 0

Voice font: mike16

Host and port: -p 7000 -s localhost

Use this voice alphabet: att_darpabet_english

Installation folder: C:\Program Files\ATTNaturalVoices\TTS4

C:\AllWork\ATT Natural Voices\Version_42_Windows\trial\client\sdk\samples\VB6-Dictionary-Editor\custom.dict

Figure 7. AT&T Natural Voices Dictionary Editor [4]

Table 1 shows the system requirements for Wizzard Software's AT&T Natural Voices SDK. This software offers numerous options for operating system support and has relatively low system memory requirements.

Table 1. Wizzard Software AT&T Natural Voices System Requirements [4]

Operating System	Windows: NT, 2000, XP, Server 2003, Vista Linux: Red Hat Enterprise Linux 5.0, 5.1, 5.2, Fedora Core 5-10, Ubuntu 8.04 GCC Version 4.1+
Memory	Windows: 256MB (128 Minimum) Linux: 512MB (128 Minimum)
Processor	Linux: 500 MHz Windows: 300 MHz
Free Disk Space	500 MB
Programming Languages Supported	C++ (Can also be integrated with Visual Basic, C#, ASP, Flash/PHP in Windows)

C. NeoSpeech

The NeoSpeech engine that can be used for developing custom TTS applications is the VoiceText™ Text-To-Speech Engine. The VoiceText engine also provides dictionary customization for pronouncing symbols, abbreviations, and new terms. The VoiceText engine was originally developed by a Korean company called Voiceware Co., Ltd. This company is the main investor for the California-based NeoSpeech company [6].

NeoSpeech offers an SDK for $950 and licenses at $550 for each computer. Several English voices are available, including Julie, Kate, and Paul. Based upon the names of the voices, it may be that the VoiceText capability is the technology being used currently on the CDT.

Limited documentation was found for the English version of the VoiceText SDK. However, an API Programmer's Guide for the Korean Engine was located [7], and Figure 8 shows some of the sample code from that document.

```
// Load the TTS Database
if (VT_LOADTTS_KOR (NULL, -1, NULL, NULL) != VT_LOADTTS_SUCCESS)
     return -1;

// Load the user dictionary
if (VT_LOAD_UserDict_KOR(1, "userdict.csv") !=
VT_LOAD_USERDICT_SUCCESS)
     return -1;

// set the pitch, speed, volume, and pause between sentence time
VT_SetPitchSpeedVolumePause_KOR (110, 90, 300, 2000, -1);
```

Figure 8. Various Code Samples from the VoiceText SDK [7]

Table 2 shows the system requirements for using NeoSpeech.

Table 2. NeoSpeech VoiceText System Requirements [6]

Operating System	Windows: 98 or higher (Vista requires administrator privilege) Linux: Red Hat Enterprise Linux 4 or higher (known to work 5.0, 5.1, 5.2, Fedora Core 5-10, Ubuntu 8.04 GCC Version 4.1+
Processor	Pentium III 500MHz
Memory	128 MB (256 MB Recommended)
Free Disk Space	64-900 MB
Programming Languages Supported	C/C++

D. Cepstral

Cepstral provides TTS technologies, and, in fact, the company claims, "Text-to-Speech is our only focus." Cepstral offers an affordable SDK at $299, and their voices are approximately $50 each. The SDK allows developers to pair their own applications with Ceptral's TTS. The SDK contains all of the .h header files and .lib libraries needed to link with a C++ project under MAC OS X, Windows, and Linux. The SDK also includes registration keys for all voices on all supported platforms to allow for uninhibited development [8].

Cepstral offers custom voice tuning that allows their technicians to alter or add phonetics that are not accessible through SSML calls. Also, the Cepstral voices are SAPI 5 compliant, so the voices can be used directly by the Microsoft Speech SDK, although the SSML capability is not supported when using SAPI instead of the Cepstral proprietary SDK.

Table 3 shows the system requirements for using the Cepstral TTS capability.

Table 3. Cepstral TTS System Requirements [8]

Operating System	Windows, Macintosh OS X, or Linux
Processor	
Memory	32 MB
Free Disk Space	10-20 MB (Per voice)
Programming Languages Supported	C/C++

Cepstral offers the use of SSML with their voices which allows the user to control the various aspects of the voice to achieve desired results. The text in Figure 9 demonstrates the method used to control the pitch of a voice.

```
"<prosody pitch='x-low'>This is half-pitch</prosody>"
"<prosody pitch='low'>This is 3/4 pitch.</prosody>"
"<prosody pitch='medium'>This is normal pitch.</prosody>"
"<prosody pitch='high'>This is twice as high.</prosody>"
"<prosody pitch='x-high'>This is three times as high.</prosody>"
"<prosody pitch='default'>This is normal pitch.</prosody>"
"<prosody pitch='-50%'>This is 50% lower.</prosody>"
"<prosody pitch='+50%'>This is 50% higher.</prosody>"
"<prosody pitch='-6st'>This is six semitones lower.</prosody>"
```

Figure 9. Cepstral's Pitch Control

Emphasis can be added to certain words to by using SSML as seen in Figure 10.

```
"This is <emphasis level='strong'>stronger</emphasis> than the rest."
"This is <emphasis level='moderate'>stronger</emphasis> than the rest."
"This is <emphasis level='none'>the same as</emphasis> than the rest."
```

Figure 10. Cepstral's Emphasis Control

E. NaturalReader

NaturalReader offers a number of voices from AT&T and NeoSpeech. They do not offer an SDK, but they offer a developer version for $199.00 which includes eight voices. This can be called by command lines from other applications to convert text to audio files. Available options are as follows [9]:

- Convert a text string into an audio file
- Convert a text file into an audio file
- Batch convert multiple text files into audio files
- Set the voice, speaking speed, and audio quality
- Converts text to MP3, WAV, or OGG audio formats

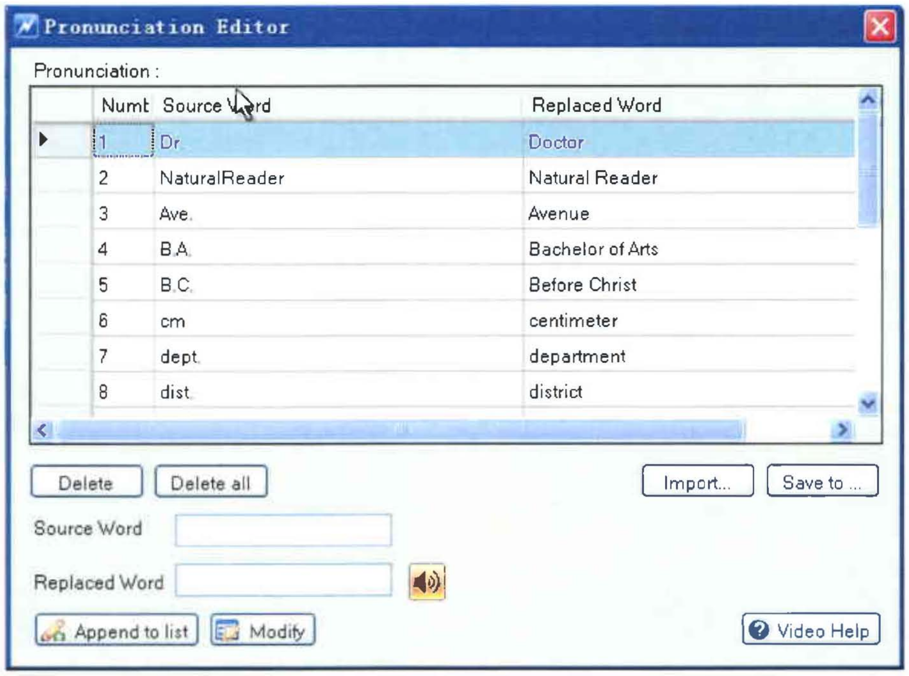
Figure 11 shows some examples for converting TTS using NaturalReader, and Table 4 provides the system requirements. Figure 12 shows the NaturalReader Pronunciation Editor, which allows the user to easily add new abbreviations and change the pronunciation of words.

```
1. Convert text into audio files:
  Convert "How are you ? I am fine." To Fine.mp3
  NaturalReaderCL -str How are you? I am fine.
     -save f:\fine.mp3 -reg XXXXXXXXXXXX

2. Convert text file into audio file:
  Convert "f:\Mfile\plan.txt" into mp3
  fileNaturalReaderCL -txt f:\Mfile\plan.txt
     -save f:\Mfile\plan.mp3 -reg XXXXXXXXXXXX
```

Figure 11. NaturalReader Examples [9]

Table 4. NaturalReader System Requirements [9]

Operating System	Windows 98/Me/NT/2000/XP/Vista/Win7
Processor	500 Mhz
Memory	128 MB (256 recommended)
Free Disk Space	50 MB (Natural voices may require 600MB free space)
Programming Languages Supported	Any language that can make a call to a command line

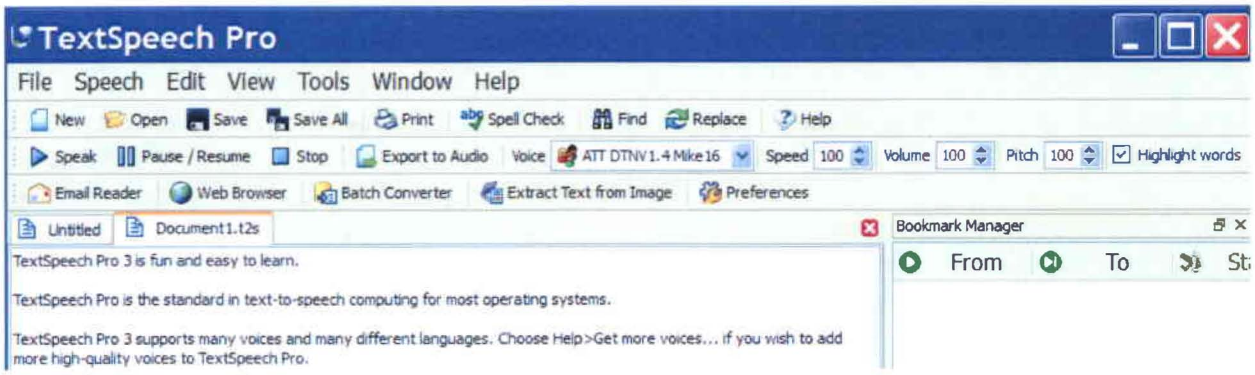


Numb	Source Word	Replaced Word
1	Dr	Doctor
2	NaturalReader	Natural Reader
3	Ave.	Avenue
4	B.A.	Bachelor of Arts
5	B.C.	Before Christ
6	cm	centimeter
7	dept.	department
8	dist.	district

Figure 12. NaturalReader Pronunciation Editor [9]

F. Digital Future

Digital Future's TextSpeech Pro application provides the capability to synthesize TTS from many document formats, such as text, Microsoft Word, rich text (RTF), and Adobe PDF. This software uses several capabilities already described in this paper, including AT&T Natural Voices, NeoSpeech, and Cepstral. An advanced editor allows the user to create conversations. Also, speech properties—such as the voice, speed, volume, and pitch—can be modified, and custom pronunciations can be applied. Command-line capability is also provided. Figure 13 shows the TextSpeech Pro application, while Figure 14 shows the Pronunciation Corrector [10].

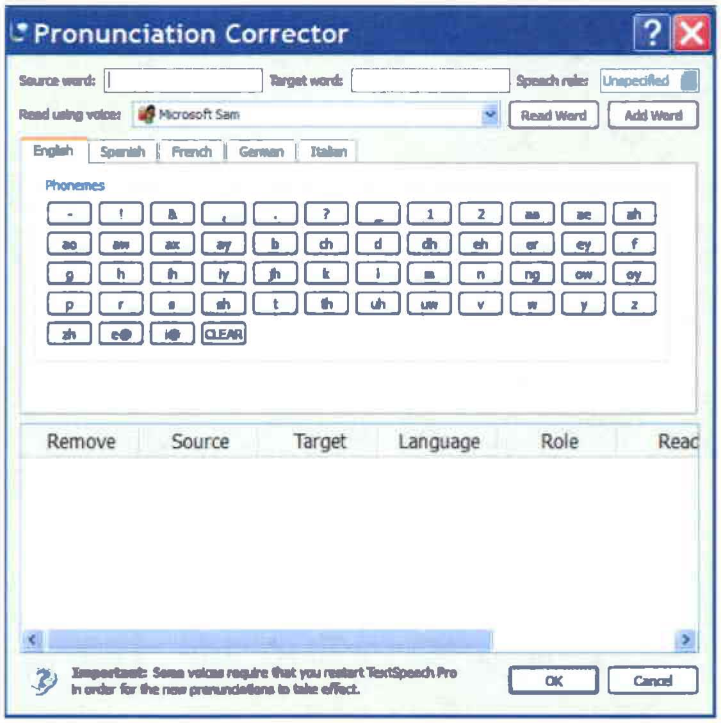

TextSpeech Pro
File Speech Edit View Tools Window Help
New Open Save Save All Print Spell Check Find Replace Help
Speak Pause / Resume Stop Export to Audio Voice ATT DTNV1.4 Mike16 Speed 100 Volume 100 Pitch 100 Highlight words
Email Reader Web Browser Batch Converter Extract Text from Image Preferences
Untitled Document1.t2s
Bookmark Manager
From To St

TextSpeech Pro 3 is fun and easy to learn.

TextSpeech Pro is the standard in text-to-speech computing for most operating systems.

TextSpeech Pro 3 supports many voices and many different languages. Choose Help>Get more voices... if you wish to add more high-quality voices to TextSpeech Pro.

Figure 13. TextSpeech Pro Application



Speech role: Unspecified

Read using voice: Microsoft Sam

Important: Some voices require that you restart TextSpeech Pro in order for the new pronunciations to take effect.

Figure 14. TextSpeech Pro Pronunciation Corrector

In addition to the TextSpeech Pro application, Digital Future also provides a TTS SDK. This SDK supports a wide variety of programming languages and provides a variety of voices. Digital Future is a cross platform application offering solutions for Windows, MAC OS X, and Linux. The programming languages supported for these operating systems are shown in Table 5. Numerous audio output formats are also available.

Table 5. Programming Languages Supported

Microsoft Windows	C++, Java, C# .NET, Visual Basic.NET, Visual Basic 6
Mac OS X	Objective-C/C++ (Cocoa), C++ (Carbon), Java
Linux	C++, Java

The Digital Future SDK provides a standard API to speech engines, and the company also provides 24-hour developer support. While not stated directly on the website, it appears as though the API interfaces to the Cepstral technology at a minimum. Figure 15 shows some example API calls. The SDK from Digital Future is available for $1500. Digital Future offers different options of voice usage and different prices for each option. If NeoSpeech voices are used the price is $100 per voice per machine. If Cepstral's voices are used, then the price is $60 per voice per machine.

```
DFTTSSpeak(0, 1, "David", -1, 1033, "Testing the Digital Future
    Text-to-Speech SDK.", -1, -1, -1, -1, -1,
    DFTTS_TEXT_TYPE_XML,-1);

DFTTSExportToFileEx(0, "Paul", 1, 1033, "Testing the Digital
    Future Text-to-Speech SDK.", -1, -1, -1, -1, -1,
    DFTTS_TEXT_TYPE_XML, "test.wav", 0, "", -1, -1);
```

Figure 15. Various Code Samples from the VoiceText SDK [9]

G. Loquendo

Loquendo is a global TTS provider that presents high quality speech synthesis. They offer a cross platform solution that supports Windows and Linux. Loquendo is an expensive alternative for TTS, since the SDK costs $2000 and there is an additional charge of $250 per application for a single voice. However, the voices are clear, natural sounding, and more realistic than any others researched for this report. Table 6 shows the system requirements.

Table 6. Loquendo System Requirements [11]

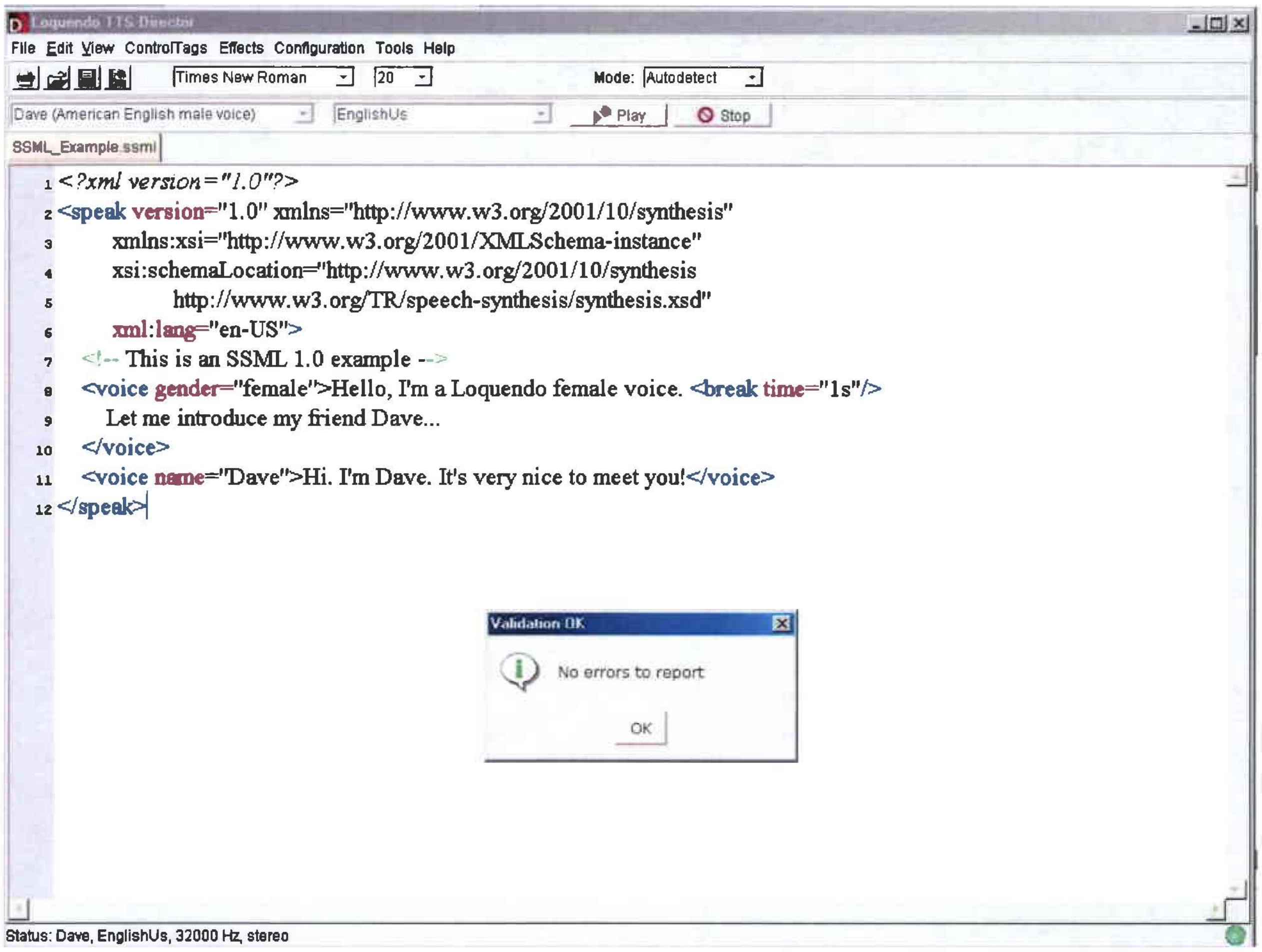
Operating System	Windows XP/Vista/Server 2003/2008 Red Hat Enterprise Linux 3, 4, 5 SUSE Linux 10
Processor	Pentium family (or equivalent)
Memory	10 MB RAM available for the engine
Free Disk Space	50 MB per voice
Programming Languages Supported	C/C++/Java

The C/C++ API provides non-object-oriented, C style interfaces into the Loquendo TTS engine. Capabilities provided through the API include setting the language, stereo balance, reverb, speed, and pitch, along with other options. While code using the API does not appear as clean as an object-oriented solution would be, it does appear fairly straightforward to use. Example calls into the Loquendo API are shown in Figure 16.

```
ttsResultType r = tts_OK;
ttsHandleType  hReader, hVoice, hLanguage;

// create a new reader
r = ttsNewReader(&hReader, NULL);

if (r != tts_OK) return r;

// create a new voice
r = ttsNewVoice(&hVoice, NULL, "Bernard");

if (r != tts_OK) return r;

// create a new language
r = ttsNewLanguage(&hLanguage, NULL, "french");

// set the voice
r = ttsSetVoice(&hReader, hVoice);

// set the language
r = ttsSetLanguage(&hReader, hLanguage);

if (r != tts_OK) return r;

// set the pitch to 55
ttsSetPitch(hReader, 55);

// set the speed to 10
ttsSetSpeed(hReader, 55);

// example of synchronous call (TTS conversion of a text file)
r = ttsRead(hReader,"input.txt",ttsFALSE,ttsTRUE,NULL);

if(r != tts_OK) return r;

// example of asynchronous call (TTS conversion of a text buffer)
r = ttsRead(hReader,"1, 2, 3, 4, 5.",ttsTRUE,ttsFALSE,NULL);

// wait until completion or a 5 sec timeout has elapsed
ttsWaitForEndOfSpeech(hReader,5000,NULL);
```

Figure 16. Code Sample Showing Use of the Loquendo SDK [11]

Figure 17 shows a screen shot of the TTS Director. This tool allows users to make changes in tone, hesitations, and various other speech qualities using SSML tags.

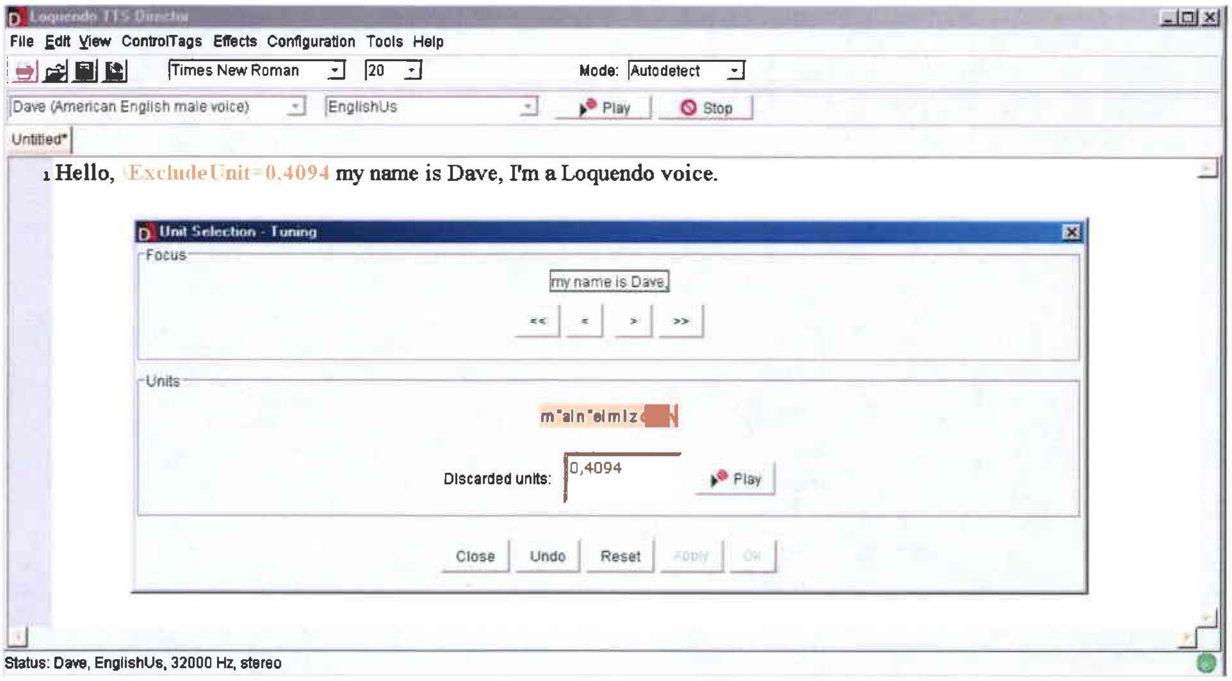


```
<?xml version="1.0"?>
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.w3.org/2001/10/synthesis
            http://www.w3.org/TR/speech-synthesis/synthesis.xsd"
      xml:lang="en-US">
   <!-- This is an SSML 1.0 example -->
   <voice gender="female">Hello, I'm a Loquendo female voice. <break time="1s"/>
      Let me introduce my friend Dave...
   </voice>
   <voice name="Dave">Hi. I'm Dave. It's very nice to meet you!</voice>
</speak>
```

Status: Dave, EnglishUs, 32000 Hz, stereo

Figure 17. TTS Director [12]

Loquendo allows the user to add realistic sounds to projects. They offer a vast repertoire of sounds such as laughs, coughs, sighs, and hesitation. If the user wants a prompt read in particular way in terms of intonation or emphasis, the User-Driven Unit Selection tool within the TTS Director software can be used (Figure 18) to obtain various alternatives for a highlighted word or words until an acceptable pronunciation is achieved.


Loquendo TTS Director
File Edit View ControlTags Effects Configuration Tools Help
Play
Stop
Untitled*
Unit Selection - Tuning
Focus
<< < > >>
Units
Play
Close Undo Reset Apply OK

Times New Roman 20 Mode: Autodetect

Dave (American English male voice) EnglishUs

1 Hello, \ExcludeUnit=0,4094 my name is Dave, I'm a Loquendo voice.

my name is Dave,

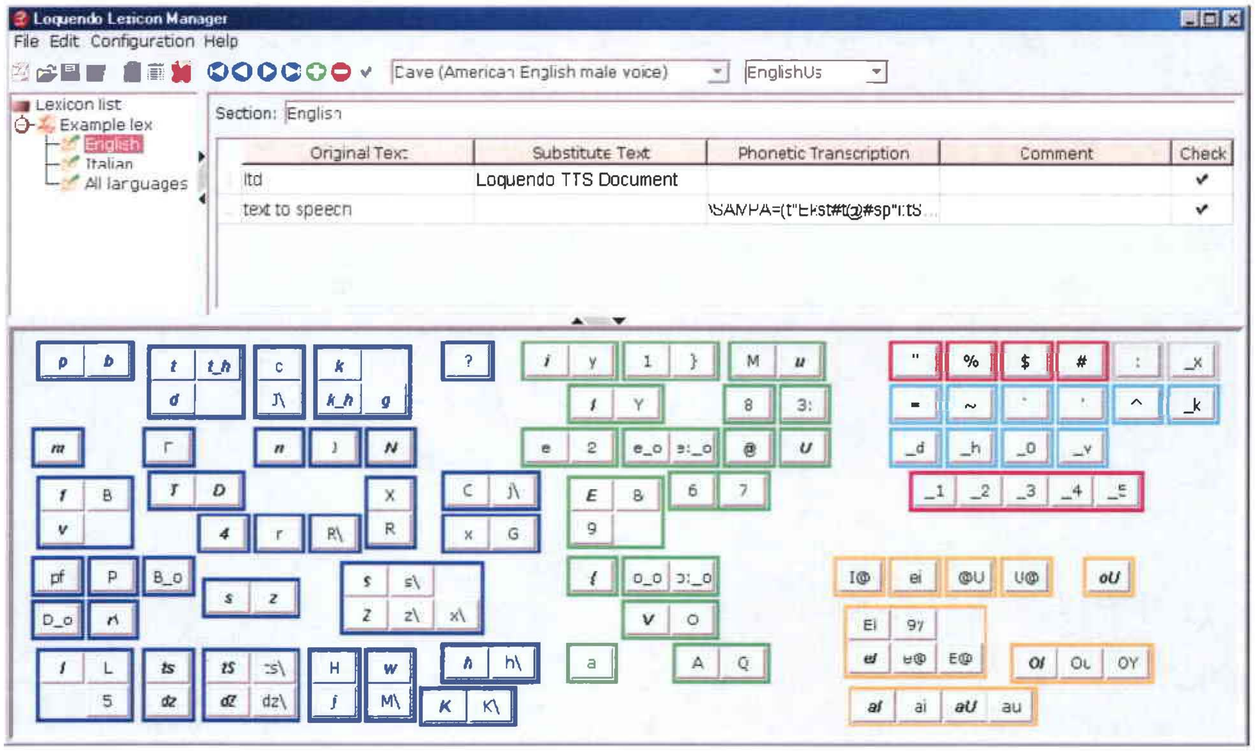
m"aIn"eImIz

Discarded units: 0,4094

Status: Dave, EnglishUs, 32000 Hz, stereo

Figure 18. TTS Director Unit Selector [12]

Figure 19 shows a Loquendo tool called Lexicon Manager, which allows the user to phonetically create the way a word is to be pronounced. Acronyms and abbreviations can be customized for speaking in a suitable manner using this tool.



Dave (American English male voice)
EnglishUs

Section: English

Original Text	Substitute Text	Phonetic Transcription	Comment	Check
ltd	Loquendo TTS Document			✓
text to speech		\SAMPA=(t"Ekst#t@#sp"i:tS...		✓

Figure 19. Lexicon Manager Tool [12]

V. SAMPLE APPLICATIONS USING MICROSOFT LIBRARIES

In preparation for the TTS research, AMRDEC developed some simple TTS applications to gain understanding of the technology. Microsoft's TTS libraries, which were described in Section IV.A, were successfully used to create a Microsoft Foundation Class (MFC) application using C++ and a Windows Forms application using C# .NET. The C# application proved to be the easiest to develop.

A. C++ Application

AMRDEC developed a C++ application that provided TTS capability using the Microsoft SAPI 5 library. A simple Graphical User Interface (GUI) was added using the MFC library. Voices can be chosen from the list of available voices installed on the computer, and the output can also be customized through setting the rate and volume.

Figure 20 shows the graphical user interface used in the C++ application. As shown, a voice can be selected from the available Microsoft SAPI-compatible voices. Also, text can be entered directly for speaking, or a text file can be provided as the text input. Figure 21 shows a portion of the code used to achieve speech synthesis in C++.

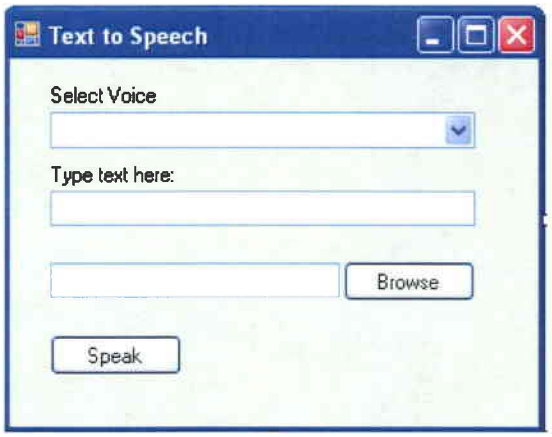


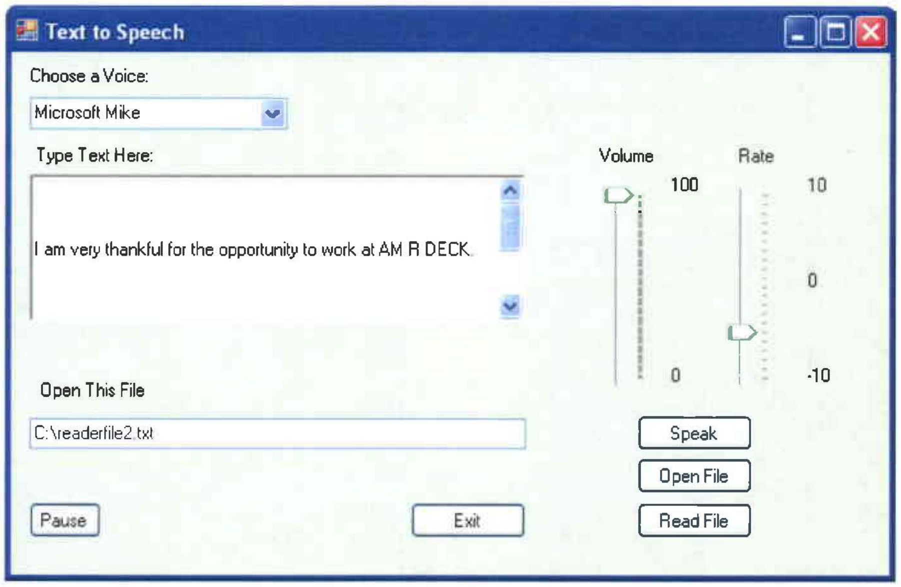
Figure 20. C++ TTS Application

```
// convert text to speech
void TextToSpeech(std::string inputText)
{
	CComPtr<ISpVoice> pVoice;

	if (FAILED(CoInitialize(NULL)))
	{
		// Error intiliazing COM
		return;
	}

	// create the voice object
	HRESULT hr = CoCreateInstance(CLSID_SpVoice, NULL,
		CLSCTX_ALL, IID_ISpVoice, (void **)&pVoice);

	CComPtr<ISpObjectToken> token;
	std::string voiceName = "Name=" + m_Voices[m_CurrentVoice];

	// find the selected voice
	hr = SpFindBestToken(SPCAT_VOICES, A2BSTR(voiceName.c_str()),
		0, &token);
	hr = pVoice->SetVoice(token);

	if( SUCCEEDED( hr ) )
	{
		BSTR bstr = A2BSTR(inputText.c_str());
		// speak the text
		hr = pVoice->Speak(bstr, 0, NULL);
		pVoice.Release();
	}

	CoUninitialize();
}
```

Figure 21. C++ SAPI Code

B. C# Application

AMRDEC also developed a Windows Forms C# application that uses Microsoft's managed code Speech API. This .NET library was found to be simpler to use than the C++ SAPI library. With a few lines of code and an integrated GUI, the developer was able to produce a working TTS application. Some of the key steps to achieving this are provided in the following paragraphs. Figure 22 shows the GUI for AMRDEC's prototype C# application.



I am very thankful for the opportunity to work at AM R DECK.

C:\readerfile2.txt

Figure 22. C# TTS Application

In order to achieve Speech Synthesis in C#, the System Speech library must first be added to the project references. Then, scope statements for the components being used must be included. At a minimum, this requires the line of code provided in Figure 23.

```
using System.Speech.Synthesis;
```

Figure 23. Scope Statements in C# TTS Application

Once the project is appropriately setup, initializing the speech synthesizing component is simple, as shown in Figure 24.

```
SpeechSynthesizer m_SpeechSynthesizer;
m_SpeechSynthesizer = new SpeechSynthesizer();
```

Figure 24. Initializing the Engine

Selecting voices from those installed on a particular computer is also easy by using the code shown in Figure 25. This code places the available voice options in a drop down box for the user's selection.

```
foreach (InstalledVoice voice in voices)
{
        string v = voice.VoiceInfo.Name.ToString();
        VoiceChoiceComboBox.Items.Add(v);
}
```

Figure 25. Adding Installed Voices to a Combo Box

The code in Figure 26 is triggered when the "Speak" button is pressed on the dialog. The action that follows is that the Speech Synthesizer speaks the text the user has typed into the text box or input from an external text file.

```
private void SpeakButton_Click(object sender, EventArgs e)
    {
        // Select the voice in the speech engine to match the
        // selected voice on the combo box
        if (Convert.ToString(VoiceChoiceComboBox.SelectedItem) != "")
        {
            m_SpeechSynthesizer.SelectVoice(
                Convert.ToString(VoiceChoiceComboBox.SelectedItem));
        }

        // Set the volume and rate
        m_SpeechSynthesizer.Volume = VolumeTrackBar.Value;
        m_SpeechSynthesizer.Rate = RateTrackBar.Value;

        // Speak the text
        m_SpeechSynthesizer.Speak(RichTextBox.Text);

    }
```

Figure 26. Speaking Text

VII. RECOMMENDATION

There are at least two design options for providing TTS capability for the CDT program, regardless of the TTS technology used. One possibility is to integrate the TTS functionality directly into the scenario-generation tool. Another approach is to create a separate application dedicated to adding or creating audio files that would then be made available to the scenario-generation system. In either case, there should be a means of generating aural cues using both TTS and recorded voice.

AMRDEC recommends that the CDT program develop an application or component that provides the following basic functions, most of which are already requirements for the CDT scenario-generation system:

- Type text into an input box for TTS creation of audio file
- Input text from a file for TTS creation of audio file
- Voice selection for the TTS engine
- Record human speech into an audio file
- Export files in .wav, .mp3, .wma formats
- Provide a custom dictionary screen that defines how the TTS engine pronounces acronyms, special words, and so forth.
- Handle file organization on the hard drive as needed by the scenario-generation system

As part of the decision process for selecting a recommended TTS capability, a matrix with various weighting factors was developed. The weights column defines the importance of each factor on a scale from 1 to 10, with 10 being most important. Each of the technologies described in Section IV were evaluated against these factors, with 10 being the best score and 0 being the worst score for a particular weight. The total score was obtained by summing the product of the weights and scores for each technology. The results shown in Table 7 are not completely scientific, since the scores are somewhat subjective and dependent upon the evaluator's understanding of the data readily available for the individual technologies. However, these results do generally show the consensus of the authors.

Table 7. Decision Matrix

Weighting Factors	Weights	Microsoft	Wizzard	NeoSpeech	Cepstral	Natural Readers	Digital Future	Loquendo
Cost (high score means low cost)	7	10	8	0	7	5	6	3
Works on Microsoft Windows	10	10	10	10	10	10	10	10
Smoothness/Natural sounding	9	5	7	9	6	8	9	10
Accuracy of pronunciation	9	7	8	7	7	7	7	10
Customizable for acronyms, pronunciation	7	5	9	8	7	2	10	10
Cross-platform	2	0	10	10	10	2	10	10
Input file formats	6	10	10	10	6	10	10	10
SDK for custom applications	4	10	10	10	10	5	10	10
Simplicity of SDK integration	3	10	9	7	5	5	5	7
Availability of COTS tool	3	0	10	0	0	10	10	10
Simplicity of COTS tool	3	0	8	0	0	10	10	9
Total Score		**443**	**555**	**441**	**426**	**443**	**551**	**569**

As shown in the decision matrix, the Loquendo library provides the best solution based upon the weighting factors considered. Selection of this library would offer the user the most natural and realistic speech out of the alternatives studied. While there will be a cost impact associated with using this software, the library is an improvement over the current CDT TTS functionality. If the cost of the Loquendo library exceeds the CDT budget for this requirement, perhaps either the Wizzard Software solution with the AT&T Natural Voices or the Digital Future TTS SDK could be utilized.

Regardless of the solution chosen, AMRDEC is capable of implementing the new software in support of the CDT program. The simplest solution would be to task AMRDEC to develop a separate application meeting the requirements stated above. However, if the TTS capability is desired to be integrated with the scenario-generation software, this, too, can be accomplished once AMRDEC is provided with the source code for that software.

REFERENCES

1. Website: www.peostri.army.mil/PRODUCTS/CDT_SV

2. PRF-PT-00430 Version 4.1, August 1, 2009, Appendix A, Scenario Generation System Specific Requirements, Section 3.5 "Sounds/Aural Cues"

3. Website: http://en.wikipedia.org/wiki/Speech_Application_Programming_Interface

4. Website: www.wizzardsoftware.com

5. AT&T Natural Voices^{TM} Text-To-Speech Engines Version 4.2 for Windows Developer's Guide, Copyright 2001-2008 by AT&T Corporation (http://www.wizzardsoftware.com/docs/att_NV_dev_windows.pdf)

6. Website: www.neospeech.com

7. VoiceText Korean Engine API Programmer's Guide, Software Version 3.7.0, VoiceText^{TM}, Voiceware Co., Ltd. (http://www.neospeech.com/manual/vt_kor-Engine-API-References-v3.7.0%20(english_translation).pdf)

8. Website: www.cepstral.com

9. Website: www.naturalreaders.com

10. Website: www.digitalfuturesoft.com

11. Website: www.loquendo.com

12. Loquendo TTS 7 Programmer's Guide, Copyright 2001-2008 Loquendo

13. Bonardo, Davide, Loquendo TTS Director: The next generation prompt-authoring suite for creating, editing and checking prompts, (www.loquendo.com/en/articles/Loquendo-TTS-Director.pdf)

LIST OF ACRONYMS AND ABBREVIATIONS

AAR	After Action Review
AMRDEC	Aviation and Missile Research, Development, and Engineering Center
API	Application Programming Interface
CIGI	Common Image Generator Interface
CDT	Common Driver Trainer
GUI	Graphical User Interface
IG	Image Generator
IOS	Instructor/Operator Station
MFC	Microsoft Foundation Class
MRAP	Mine Resistant Ambush Protected
SAPI	Speech Application Programming Interface
SDK	Software Developer Kit
SRGS	Speech Recognition Grammar Specification
SSML	Speech Synthesis Markup Language
TTS	Text to Speech

INITIAL DISTRIBUTION LIST

		Copies
Weapon Systems Technology Information Analysis Center Alion Science and Technology 201 Mill Street Rome, NY 13440	Ms. Perry E. Onderdonk ponderdonk@alionscience.com	Electronic
Defense Technical Information Center 8725 John J. Kingman Rd., Suite 0944 Fort Belvoir, VA 22060-6218	Mr. Jack L. Rike jrike@dtic.mil	Electronic
AMSAM-LI	Ms. Anne C. Lanteigne anne.lanteigne@us.army.mil	Electronic
RDMR		Electronic
RDMR-CSI		Electronic
RDMR-WDG	Dr. Robin Buckelew robin.buckelew@us.army.mil	Electronic
	Mr. Jim Hatfield jim.hatfield@us.army.mil	Electronic
	Ms. Julie Locker julie.locker@us.army.mil	Electronic
RDMR-WDG-C	Mr. Roger Berry roger.berry@us.army.mil	Electronic
	Mr. Danny Carter danny.carter1@us.army.mil	Electronic
	Mr. Michael Hanners michael.hanners1@us.army.mil	Electronic/Hardcopy
PEO STRI SFAE-STRI-PS-E-S 12350 Research Parkway Orlando, FL 32826-3276	Mr. Darryl Williams darryl.williams2@us.army.mil	Electronic
PEO STRI SFAE-STRI-PSG-E-S 12350 Research Parkway Orlando, FL 32826-3276	Mr. Dean Runzel dean.runzel@us.army.mil	Electronic