

## **An introduction to open source and low-cost text-to-speech technologies**

**Robin M Snyder**  
**robin@robinsnyder.com**  
**<http://www.robinsnyder.com>**

### **Abstract**

Traditionally expensive to acquire and license, increasingly lower cost voices are becoming available and have spawned many open source and low cost technologies for text-to-speech applications. The author/presenter has recently done extensive commercial research and software development work in text-to-speech technologies. This paper/session will cover background, working examples, and applications of text-to-speech technologies that can be used for student projects, classroom teaching, and institutional computing applications.

### **Introduction**

This paper/session will cover background, working examples, and applications of text-to-speech technologies that can be used for student projects, classroom teaching, and institutional computing applications.

### **Background**

Sound is created from vibrations in a medium, usually air, that is transferred from a source creating the sound to a detector that can recognize the sound. A sound recorder will save the sound as an audio signal. A human ear will pass the sound on to the brain for recognition and possible understanding.

We will not be concerned with philosophical discussions of the form, "If a tree falls in a forest, does it make a sound". Nor will we be concerned with defining information as that leads to metaphysical implications that may or may not be of interest to the reader.

The issue here is that of speech to text. Speech to text is the conversion of sound via audio to textual form (i.e., the textual syntax). Speech recognition is the conversion of spoken audio to textual form such that the meaning of the text is clear (i.e., the textual semantics).

Obviously the speech recognition problem is much harder than speech to text problem. Any text to meaning problem suffers from the inherent ambiguities of the language being used. Example: The man saw the woman with a telescope. Who has the telescope? Natural language is inherently ambiguous. As then President Bill Clinton testified, "It depends on what the meaning if is is".

This paper is concerned with the automatic conversion of text to voice using open source and low-cost technologies. Some ending comments will be made on text to singing voice.

The text to voice problem is easy to solve such that people can understand the text to audio. This has been done for years with text readers to greatly benefit deaf or hard-of-hearing people. But it

does not help very well with hard-of-listening people - the Dilbert managerial approach recommended as "if they don't understand, repeat yourself slower and louder". My mother often said about her children that "You all can hear well, you just don't listen."

However, the text to voice problem is far from being solved such that the voices sound very natural. Since the first full length animation movie Toy Story appeared, computer animation has become more and more life-like. EA Sports with their motto of "If it's in the game, it's in the game" has been producing sports games that are more and more realistic.

The movie and game industry would greatly like to not pay Tom Hanks, etc, millions to do the voice tracks for their animations, but, unlike animation, the technology is not ready for totally realistic voice effects.

There are many text-to-voice demos on the Internet. An Internet search of the terms "voice text demo" should suffice to find many of them.

## Microsoft TTS

A starting point for text to voice conversion is the Microsoft TTS (Text To Speech) technology since it is freely available with Windows and/or with additional downloads.

The disadvantage is that the voices sound mechanical and robotic.

Here is a simple Visual Basic script program to say "**Hello, World**".

```
Dim voice1
Set voice1 = CreateObject("SAPI.SpVoice")

voice1.Speak "Hello world."
voice1.WaitUntilDone(0)
Set voice1 = Nothing
```

To run the program, save it as a text file called **hello1.vbs** and run it. Here are command lines to run it.

```
wscript.exe hello1.vbs
hello1.vbs
```

Here is the same program expanded to save the spoken text to a file.

```
Dim voice1
Set voice1 = CreateObject("SAPI.SpVoice")

Dim stream1
Set stream1 = CreateObject("SAPI.SpFileStream")

Dim fs1
fs1 = "D:\hello1.wav"

stream1.Open fs1,3,false
Set voice1.AudioOutputStream = stream1
voice1.Speak "Hello world.",0
voice1.WaitUntilDone(0)
stream1.Close
Set stream1 = Nothing
Set voice1 = Nothing
```

A similar program can be written in JavaScript and run at the client browser but the user must be running Windows, have enabled or enable the appropriate ActiveX control when requested, etc. Such security restrictions and limited browser support make this an unattractive way to play sound at the browser.

### **Flash/Flex/ActionScript**

A fairly portable way to play the audio at the web browser in a seamless way (i.e., without downloads, plug-ins, etc.) is to use a Flash component that communicates to/from with the web page via JavaScript and user interaction and communicates to/from the web server via XML transfers and other direct transfer techniques (e.g., for direct audio transfer).

Developing Flash components in Flash can be very difficult for programmers. Flash was designed by artists for artists and not by programmers for programmers. Many ways of doing things in Flash apparently make sense to artists but not to programmers. However, Adobe has packaged the ability to create Flash components in what is called Flex. Flex is much more programmer friendly and is based on the web page model with support for CSS. Behind the scenes, both Flash and Flex use ActionScript as a JavaScript-based object-oriented programming system.

What may not be obvious is that one can download the Flex system SDK (Software Development Toolkit) from Adobe for free (unlike Flash) and start programming text-based Flex programs using ActionScript that produce Flash animations and/or graphical user interfaces that can be embedded in web pages.

Debugging Flex/ActionScript can be difficult until one installs the debug version of Flash. All debugging text is appended to a text file. The use of an open source program like Tail can be used to reread that text file every time it changes.

Many web sites that use Flash components often leave debugging trace information in their components and it can be interesting to view such debugging information while visiting their web site.

Flash components provide the ability to store cookies without the limitations of standard cookies which may or may not be a security/privacy concern.

Like most programming systems used to develop graphical interfaces, Flex programs are not trivial for even the "**Hello, World**" program.

Here is how a Flash/Flex component, written by the author and outlined by the thin black line, appears.

Select a voice/recording to play and parameters to modify that voice.

Create new speech file at the server even if already cached.

**Remaining syntheses: 184 (refresh page to update)**

Tempo change: very slow, slower, none, faster, very fast

Pitch change: very low, lower, none, higher, very high

Special effects: none, Dath Vader, Chipmonk, Echo

Click the voice to play the recorded/modified audio for the joke "Two peanuts walk into a bar. One was asalted":

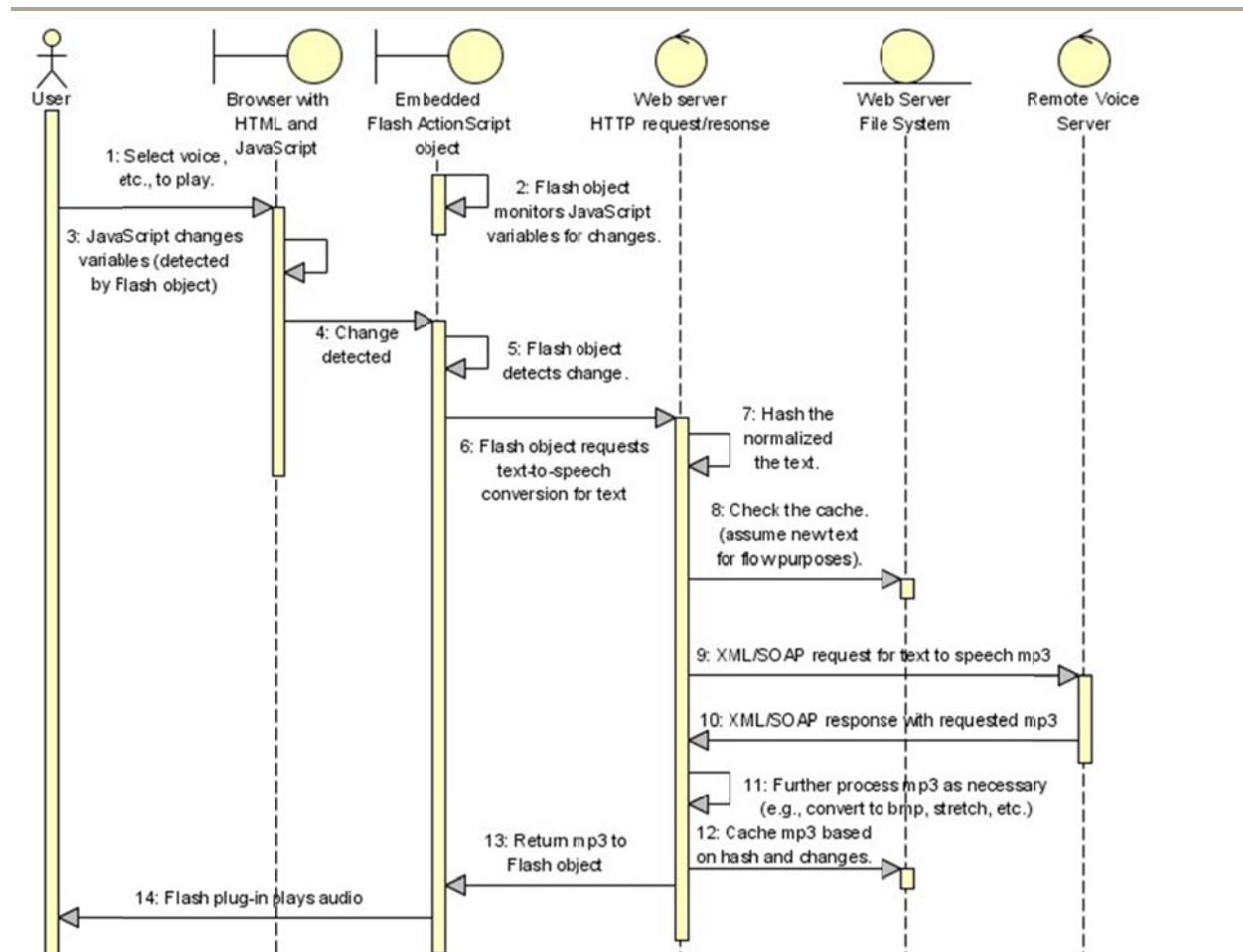
001-english	006-jamaica	011-italian
002-cogney-british	007-valley	012-british-uppity
003-brooklyn	008-jock	013-irish
004-german	009-normal	
005-texas	010-french	

It is part of a larger web page that uses JavaScript to/from Flash/Flex/ActionScript communication as well as Flash/Flex/ActionScript communication to/from the web server.

There are various tempo changes, pitch changes, and special effects that can be done on the sound. Some sounds come from the voice server while some are recorded as spoken by a high school actress with a talent for accents and voice inflections.

### Web model

The web model used is that the user will hear text converted to voice using a web browser. The web server may need to access other resources to do the conversion and may then do local conversions on the sound. Here as a UML sequence diagram for one way to do this.



Once the infrastructure is in place, the audio acquisition can be done from the voice server while additional audio processing and transformation can be done at the web server.

To do such processing and transformations a large number of open source audio processing systems can be used.

Note that, in general, open source programs can be used at the web server with significant restrictions as long as the user at the browser is not directly using the system but, instead, going through the web server as a go-between.

### Audacity and Nyquist

The open source audio processing program Audacity has macro support in the form of the Nyquist programming language - which can be used without Audacity. Here is a Nyquist program to load and play a sound file.

```

(display "File: play1.lsp")
(display "Purpose: Load and play a sound file")
(setq fs1 "D:\\W\\BT.WAV\\mary0.wav")
(play-file fs1)
(exit)

```

Here is the output of the above program.

```
File: play1.lsp :
Purpose: Load and play a sound file :
44880 88740 132600
total samples: 173056
```

Nyquist does not appear to support MP3 files. Since the patent expiration of the mp3 file format, Audacity now supports mp3 file format conversion directly using the open source lame encoder/decoder.

## Programming systems

Here are some programming systems for audio processing.

- 1. LilyPond Scheme (audio processing)
- 2. Nyquist LISP (audio processing)
- 3. Snack Sound Toolkit (audio processing)
- 4. Praat (phonetics)

## Language/notation standards

Here are some languages and standards that are useful for audio processing.

- 1. WSDL - Web Services Description Language
- 2. SSML - Speech Synthesis Markup Language
- 3. Music markup notations (XML)
- 4. Microsoft SAPI
- 5. VoiceXML
- 6. IPA phonemes
- 7. Prosody (SSML support)

## Command line tools

Here are some open source command line tools that are useful for processing audio at the web server.

- 1. SoundStretch (change WAV tempo/pitch)
- 2. LAME Encoder (convert WAV to MP3)
- 3. Mp3splt (split MP3)
- 4. SoX: Sound eXchange (convert formats, pitch/tempo)
- 5. Div's MIDI utilities (MIDI)
- 6. C# wave editor (WAV properties to XML)
- 7. LilyPond command line tools (open source)
- 8. eSpeak (open source text-to-speech)
- 9. Madplay.exe (convert MP3 to WAV)

## **GUI tools**

Here are some GUI tools for audio processing.

- 1. Audacity (free audio acquisition and processing)
- 2. MDVP - Multi-Dimensional Voice Program (commercial voice analysis)
- 3. Syng2 plugin for SynthEdit
- 4. Nuance Cafe (telephone-based speech applications)
- 5. Yamaha Vocaloid2 (commercial singing synthesizer application)
- 6. NaturalSoft NaturalReader (commercial Windows desktop)
- 7. Microsoft TTSApp demo for SAPI
- 8. Microsoft Narrator (Windows screen reader)
- 9. Audio Level Meter (sound levels)
- 10. Sound recorder

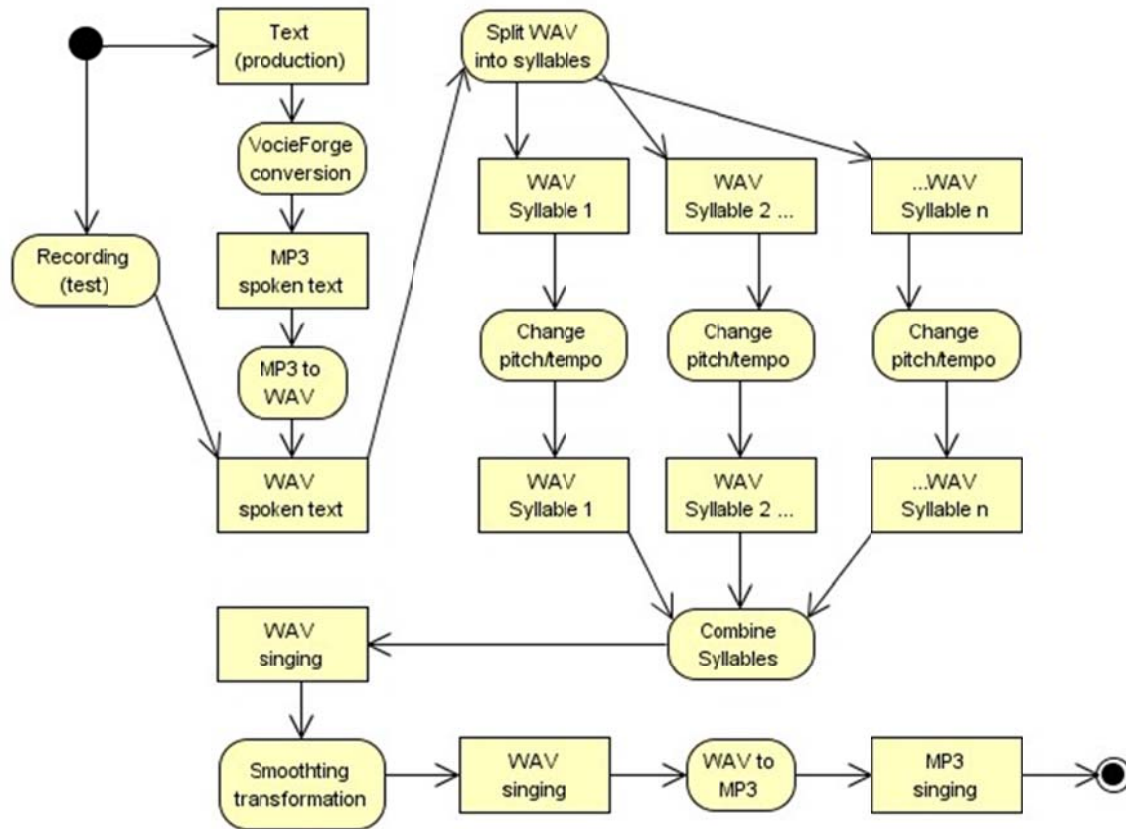
## **Converting text to a singing voice**

Part of the research involved converting text to a singing voice. Not a great singing voice, but a singing voice. A problem statement is the following: Given the melody and words of a song, convert the words to audio text and then to what sounds like the voice singing the song.

The approach was the following.

- Use system like LilyPond to specify the melody and word synchronization. A MIDI note sequence is generated.
- Convert text to MP3 word by word.
- Stretch words to the proper time
- Change the pitch of the combined words to the proper pitch.
- Merge and smooth the resulting audio.

The UML for this approach is as follows.



LilyPond will take a simple text specification of a song and create an image using music notation. Here is an example.

## Mary Had a Little Lamb



The main obstacle in the above approach was in that the voice server did not support certain breaks in VoiceXML that would have allowed the splitting of the generated voices. The company providing the voice conversions stated that it was not possible even though I presented them with a simple XML mode (and way to implement it) that would have made it possible.

The approach was implemented using the Microsoft voices (not the best voice quality) and with manually-split voices recorded with Audacity (not amenable to automation).

### **Summary**

This paper/session has covered background, working examples, and applications of text-to-speech technologies that can be used for student projects, classroom teaching, and institutional computing applications.

### **References**

- [68] Snyder, R. (1999). Using children's songs to teach abstraction techniques in an introductory programming course *The Journal of Computing in Small Colleges*. Vol. 14. No. 2. pages 92-100.
  
- [117] Snyder, R. (2003). Getting started with the UML and round-trip engineering using Rational Rose 34th Annual Meeting of the Decision Sciences Institute (November 22-25, 2003), Washington, DC. CD.
  
- [138] Snyder, R. (2005). A UML specification for a secure XML-based transfer of data using an intermediate server in an e-commerce system 41th Annual Meeting of the Southeastern Chapter of the Institute for Operations Research and the Management Sciences (October 6-7, 2005), Myrtle Beach, SC. pages 375-384.