SpeechRecorder – a Universal Platform Independent Multi-Channel Audio Recording Software

Christoph Draxler, Klaus Jänsch
Institut für Phonetik und Sprachliche Kommunikation
Ludwig-Maximilians-Universität München
Schellingstr. 3, D 80799 Munich
{draxler|klausj}@phonetik.uni-muenchen.de

Abstract

SpeechRecorder is a platform independent audio recording software for speech corpus recordings. It is implemented in Java in a clean object-oriented design and adheres to established technology standards and document interchange formats. SpeechRecorder allows Unicode text and multimedia prompts, it supports audio recordings via more than two channels, and it features multiple configurable screens. Recording sessions are defined by recording scripts written in XML. The recording scripts can be executed manually by the experimenter, or automatically for unsupervised recordings; progress through the script can be sequential or randomized. SpeechRecorder is based on URLs to access local and network resources and thus allows recordings via the WWW.

1. Introduction

Todays PCs and laptops are equipped with built-in hardware for playing and recording high quality audio. Additionally, there are many audio boards and devices available that extend the capabilities of the built-in hardware, e.g. by providing support for higher sample rates or better quantization, more input channels, or digital I/O.

The audio hardware is accessed via vendor-specific drivers. These drivers provide an interface to standard and potentially platform independent audio libraries, e.g. Windows Media, QuickTime, or Java Sound API. High-level audio applications such as recording tools, annotation editors, or audio players are based on these audio libraries.

Most commercial audio recording software is tailored to the requirements of either musicians or signal processing engineers. As a consequence, this software does not address the specific needs of recordings for speech corpora.

Speech recordings, be it field or studio recordings, require

- sophisticated prompt display,
- flexible recording scripts,
- a choice of different input channels,
- visual and acoustical signal output, and
- an efficient and intuitive user interface.

There are a number of speech recording tools available, but in general they are platform dependent, they are inflexible ad-hoc solutions for a particular recording task, or have severe technological limitations such as text only prompt display or at most stereo recordings (e.g. [4], [5], [7], [8], [9]).

The paper is structured as follows: Chapter 2 gives an overview of the architecture and the key features of SpeechRecorder. Chapter 3 describes recording scripts. Chapter 4 discusses the audio library developed at our institute that provides an additional abstraction layer above the audio libraries and thus significantly reduces the complexity of accessing these audio libraries. Finally, chapter 5 presents projects in which SpeechRecorder has been successfully used, and chapter 6 gives an outlook on future work, especially performing recordings via the WWW.

2. SpeechRecorder

SpeechRecorder is a modern speech recording software implemented in Java for platform independence. It was implemented from scratch in a clean object-oriented design. Its main features are:

- any number of audio input channels
- Unicode text, image or audio prompts for the speaker
- XML formatted recording scripts and speaker databases
- support for multiple screens
- support for internationalization.

Access to resources is achieved via URLs. As a consequence, audio files can be saved to local disks or network locations.

2.1. Graphical user interface

The graphical user interface of SpeechRecorder distinguishes between two types of screen displays: a speaker screen and an experimenter screen. The speaker screen consists of a recording indicator (don't speak, prepare to speak, speak), and two panels for the recording instructions and the recording prompt (Figure 1).

Figure 1. Speaker screen

The experimenter screen contains the speaker display in one panel, plus a recording level meter, a graphical signal display, a recording control button panel, a speaker database window and a recording script progress panel. This progress panel lists all items of a recording session. It is updated whenever an item is recorded, and the

experimenter may interactively select an item by clicking on the appropriate entry in the list (Figure 2).

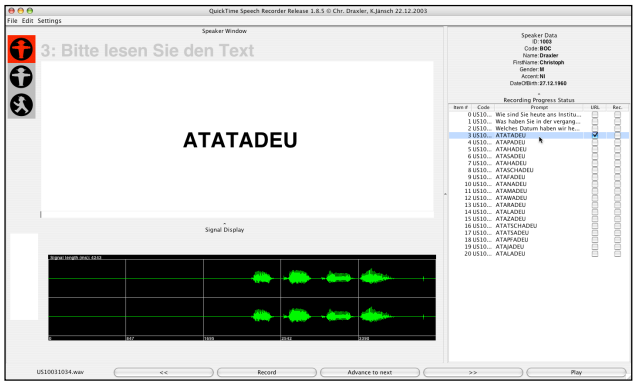

Figure 2. Experimenter screen

All interface text is stored in external Unicode property files so that SpeechRecorder can be localized easily and without recompiling the application. Currently, resource files exist for German, English, French, Russian and Greek.

2.2. Architecture

The software consists of five major components.

A *recording script manager* to load and execute a recording script. This manager keeps track of the recorded items, it selects the next recording to be performed, and progresses through the script. The selection can be sequential or randomized, and progress through the script can be manual, e.g. by an experimenter clicking on the recording control buttons, or automatic, e.g. for unsupervised recordings.

A *recording status class* that represents the status of the current recording item. A recording can be in different states: idle, waiting, recording, processing, etc. All components registered with the status object are notified of a status change.

Prompt viewers present the current prompt item to the speaker. A prompt item consists of instructions to the speaker, e.g. "Spell the following word", a free-form comment field visible only to the experimenter (e.g. to display pronunciation hints in IPA or SAM-PA), and the recording prompt proper. This recording prompt can be any Unicode text, an image or a pre-recorded or synthesized audio signal.

A *recording control panel* contains buttons to start and stop recordings, to play the current audio signal, and to step through the recording script.

The *SpeechRecorder* class is responsible for running the application: it holds references to the prompt source, the speaker database and the target directory for the recordings. All these resources are addressed via URLs so that they can be held statically in local files or web pages, or generated dynamically by a web server. For each audio file, a document is generated to store the descriptive data for this particular recording, e.g. recording time, date, duration, prompt text or description, etc.

The entire application is highly configurable, with the configuration data held in external XML files.

2.3. System requirements

SpeechRecorder requires Java 2 version 1.4. or newer for the XML handling APIs and a stable implementation of the Java Sound API libraries (cf. section 4). The software has been successfully run under Windows 2000 and XP, Mac OS X and Linux.

3. Recording scripts

For every recording session there must exist a recording script. A recording script is an XML file that consists of a meta-data section and the script according to the DTD given in Figure 3.

```
<!ELEMENT session
 (metadata*, recordingscript)>
<!ATTLIST session id CDATA #REQUIRED>

<!ELEMENT metadata (key, value)+>
<!ELEMENT key (#PCDATA)>
<!ELEMENT value (#PCDATA)*>

<!ELEMENT recordingscript
 (nonrecording | recording)+>
<!ELEMENT nonrecording (mediaitem)>

<!ELEMENT recording
 (recinstructions?,recprompt,reccomment?)>

<!ATTLIST recording
 file CDATA #REQUIRED
 recduration CDATA #REQUIRED
 prerecdelay CDATA #IMPLIED
 postrecdelay CDATA #IMPLIED
 finalsilence CDATA #IMPLIED
 beep CDATA #IMPLIED
 rectype CDATA #IMPLIED>

<!ELEMENT recinstructions (#PCDATA)>
<!ATTLIST recinstructions
 mimetype CDATA #REQUIRED
 src CDATA #IMPLIED>

<!ELEMENT recprompt (mediaitem)>
<!ELEMENT reccomment (#PCDATA)>
<!ELEMENT mediaitem (#PCDATA)*>
<!ATTLIST mediaitem
 mimetype CDATA #REQUIRED
 src CDATA #IMPLIED
 alt CDATA #IMPLIED
 autoplay CDATA #IMPLIED
 modal CDATA #IMPLIED
 width CDATA #IMPLIED
 height CDATA #IMPLIED
 volume CDATA #IMPLIED>
```

Figure 3. recording script DTD

The `metadata` section contains attribute-value pairs describing the current recording project (place of recording, project name, institution, equipment used, etc.), recording data (sample rate, quantization, format, number of channels, etc.) and which are the same for all recording sessions.

The recordingscript element consists of non-recording and recording items. Non-recording items are used to pass information to the speaker without recording speech, e.g. welcome and warning messages, or animations to temporarily distract the speaker during long recording sessions.

Recording items consist of optional recording instructions and comments, and a mandatory prompt item. The required recording attributes define the file name under which a recording is stored and the duration of the recording in milliseconds. The remaining recording attributes are optional: prerecdelay specifies how much earlier the real recordings begin before the recording start is being signaled to the user. This is used to record environment noise or to capture barge-in. postrecdelay specifies how long the recording continues after the stop button has been pressed. This is useful to prevent signal truncation due to pressing the stop button too early. finalsilence indicates whether recordings can be terminated by pause detection, beep specifies whether a recording is preceded by an indicator beep to the speaker. rectype is reserved for future use; it will specify whether audio or video recordings are performed.

Non-recording items, recording instructions and recording prompts are of type mediaitem, i.e. they are described by a mandatory mimetype attribute and optional attributes appropriate for the given type, e.g. volume for audio or video prompts, or width and height for image and video prompts.

A sample recording script is given in Figure 4.

```
<?xml version="1.0" encoding="UTF-8"
  standalone="no" ?>
<!DOCTYPE session SYSTEM
"SpeechRecPrompts.dtd">

<session id="BITS Syntheseaufnahmen 2004">
 <metadata>
  <key>DB_Name</key>
  <value>BITS Synthesekorpus 2004</value>
 </metadata>

 <recordingscript>
  <nonrecording>
   <mediaitem mimetype="image/jpeg"
              src="willkommen.jpg"/>
  </nonrecording>

  <recording recduration="6000"
             file="US10031034.wav"
             prerecdelay="500"
             postrecdelay="500">

   <recinstructions mimetype="text/UTF-8">
    Bitte lesen Sie den Text
   </recinstructions>

   <recprompt>
    <mediaitem mimetype="text/UTF-8">
     ATATADEU
    </mediaitem>
   </recprompt>

  </recording>
 </recordingscript>
</session>
```

Figure 4. Recording script

4. Audio in- and output

Access to audio hardware is implemented in several layers: on the lowest level, device dependent drivers establish an interface between the audio device and audio libraries. On this driver layer, the best-known standards are SoundBlaster, ASIO (Audio Stream I/O), and ALSA (Advanced Linux Sound Architecture) drivers. ASIO has become the de-facto standard for professional audio boards with multi-channel digital I/O, e.g. Digidesign, M-Audio, etc. The ASIO drivers can be directly accessed from programming languages such as C and C++ [10].

On top of the driver layer are the audio and multimedia libraries such as Windows Media, QuickTime, Java Sound API, or operating system libraries.

Windows Media is the multimedia library of the Windows operating system. It is not available for other platforms.

QuickTime is a library for playing, capturing and editing multimedia content. QuickTime for Java is an API to access the QuickTime libraries from Java. However, because QuickTime is only available for Macintosh and Windows, the full platform independence of Java cannot be achieved.

The Java Sound API is a platform independent low-level audio library. Currently, Windows, Solaris, Linux and Macintosh are supported. However, ASIO drivers are not recognized by the standard implementation of the Java Sound API in J2SE, so that accessing professional audio boards required expensive and proprietary solutions. The Java Media Framework, a full multimedia library, uses the Java Sound API.

4.1. IPSK audio library

For SpeechRecorder, the audio library ipsk.audio was designed. It provides extensions to Java Sound API both on the level of device drivers and on the application programming interface level.

Its distinguishing features are

- Java wrappers for the ASIO drivers, an
- abstraction from the low-level details of the Java Sound API library, and
- full support for resource access via URLs.

The ASIO drivers support recordings via more than two channels, digital I/O and higher sample rates and quantization. The Java wrapper classes in ipsk.audio provide a seamless integration of these drivers into the Java Sound API.

The abstraction from the underlying audio libraries is achieved by providing interface definitions tailored to the requirements of speech recordings. This abstraction simplifies the implementation of audio recording applications, and it allows exchanging audio libraries against each other.

In ipsk.audio, external media resources are referred to via URLs.

The ipsk.audio library is also used in WebTranscribe, a Java re-implementation of the web based annotation software developed at our lab [2].

4.2. `ipsk.audio` in SpeechRecorder

In SpeechRecorder, access to audio libraries is provided by the interface class `AudioController`. `AudioController` provides methods to open and close a recording session, to record and playback audio signals, and to convert audio file formats.

Currently, two implementations of the interface exist. The first is based on the Java Sound API, the second on QuickTime for Java.

Which audio library to use for a given recording session is specified in the configuration file. When the application starts up, it dynamically loads the appropriate implementation of `AudioController`.

5. Projects using SpeechRecorder

Early versions of SpeechRecorder have been used in several smaller in-house projects. The experiences gained led to an improvement of the software in terms of reliability, user friendliness, and features.

In the spring of 2003, IPSK recorded non-scripted speech in cars in a project for Bosch GmbH. The speakers, who were driving the car, were prompted via pre-recorded audio prompts so that they could focus on the street and did not have to read text from a display. The experimenter on the co-driver seat manually proceeded through the recording session. For these recordings, SpeechRecorder was installed on a Sony VAIO running Windows 2000 and a Macintosh PowerBook G3 running Mac OS X. Two AKG mouse microphones were used, the microphone amplifiers were L&H Mouse Amp 31. The microphones were connected to the laptop via an iMic USB audio adapter.

In July 2003 the IPSK performed recordings for the IPA in St. Petersburg, Russia [1]. SpeechRecorder was installed on a Macintosh iBook with the microphones connected to the laptop via an Emagic 62m USB audio interface.

Since the summer of 2003, the BITS speech synthesis corpus is being recorded at the Bavarian Archive for Speech Signals BAS (`www.bas.uni-muenchen.de/Forschung/BITS`). For these recordings, the prompts are logatomes and phonetically rich sentences. Two microphones (Beyerdynamic NEM 192 and Neumann Type TLM) and a laryngograph (PCLX) unit are used. They are fed via a digital Yamaha O2R sound mixer into an M-Audio digital audio board in a PC running under Windows XP [3].

6. Future work

SpeechRecorder has reached a stable state now. We will perform a public beta test during the summer of 2004 and make the software publicly available later in the year 2004 [6].

Currently, SpeechRecorder is a standalone Java implementation. It is planned to convert it into a Java WebStart application so that it can be run directly from the web.

Finally, SpeechRecorder will be extended to allow recordings via the web. This poses several new challenges: 1) slow connections will cause long uploads to the file server, and care has to be taken that this transfer is reliable. 2) the audio equipment on the client machine cannot be known in advance. Basic signal processing will have to be performed on the client machine to achieve a minimum standard of signal quality. 3) User privacy and secure transfer of sensitive data will have to be considered.

A prototype implementation of SpeechRecorder with web recording has been successfully tested at IPSK.

7. Acknowledgments

Parts of this work have been supported by the German Federal Ministry of Education and Research grant no. 01IVB01 (BITS).

8. References

[1] Dioubina, O., Pfitzinger, H.: Illustration of the Russian language in IPA, to appear.

[2] Draxler, Chr.: WWWTranscribe - a modular transcription system based on the world wide web, Proc. of Eurospeech 95, Rhodes

[3] Ellbogen, T.; Steffen, A.; Schiel, F.: The BITS Speech Synthesis Corpus for German, in: Proc. of LREC 2004, Lisbon

[4] Jameton, O.; Fonollosa, J. A. R.; Richard, G.; Comeyne, R.; Realisation of the "in-car" recording platform and installation of the fixed recording platform; internal report SD212V24; SpeechDat-Car project LE4-8334; 1999

[5] Audacity: audacity.sourceforge.net

[6] SpeechRecorder: www.bas.uni-muenchen.de/Bas

[7] PeriSpeech: www.nortelnetworks.com

[8] Praat software: www.praat.org

[9] Quikscribe: www.quikscribe.com

[10] ASIO SDK: www.steinberg.net