Submission Cover Sheet for 1999 IEEE SoutheastCon

Submission Type (Please Check One):

☐ ***Full Length Paper - (Due 11/1/98)*** [✓ ***Concise Paper - (Due 12/1/98)***]

Title of manuscript:
A UNIX-BASED SPEECH DATA COLLECTION PLATFORM

Authors and affiliations (List authors in the order to appear on the manuscript. List affiliations (companies, universities, etc.) and addresses exactly as they should appear. If there are more than 4 authors, please attach a second sheet.

1. Name	**Richard Duncan**	3. Name	
Affiliation	**Mississippi State University**	Affiliation	
Address	**PO Box 9571**	Address	
	Mississippi State, MS 39762		
2. Name	**Dr. Joseph Picone**	4. Name	
Affiliation	**Mississippi State University**	Affiliation	
Address	**PO Box 9571**	Address	
	Mississippi State, MS 39762		

Corresponding Author:

Name:	**Richard Duncan**
Address:	**PO Box 9571**
	Mississippi State, MS 39762
Telephone:	**601-325-8335**
FAX:	**601-325-3149**
E-mail:	**duncan@isip.msstate.edu**

Key Words: **speech data collection, digital telephony, application design tool**

For Technical Committee Use Only -- Do Not Write Below

Date Received:		Manuscript # Assigned	
Author Notification Date:		Date Camera Ready Received:	
Decision:		Session:	

Notes:

A UNIX-BASED SPEECH DATA COLLECTION PLATFORM

Richard Duncan, Joseph Picone
Institute for Signal and Information Processing
Mississippi State University
Mississippi State, Mississippi 39762, USA
e-mail: {duncan, picone}@isip.msstate.edu
Ph (601) 325-3149 - Fax (601) 325-3149

Abstract - **It is highly desirable to collect speech data from the telephone network via a digital interface. This avoids an additional A/D conversion normally required by analog telephone data collection hardware. A popular solution to this problem is the use of a T1 line which offers 24 digital phone lines. The leading T1 interface for Sun workstations is a system developed by Linkon Corporation. Using the Linkon framework, we have developed a fully-expandable, robust environment for platform-independent collection of telephone speech data. The object-oriented software libraries and intuitive GUI provide powerful tools with which even a novice user can efficiently prototype complex applications. The system is currently being deployed by the Linguistic Data Consortium to collect part of the next SWITCHBOARD Corpus.**

INTRODUCTION

The first system featuring a digital interface that was deployed for full scale data collection in speech research was the T1-based system built on the Intervoice Robot Operator hardware platform. This environment featured an IBM PC with an interface consisting of two proprietary boards, an OS/2 operating system (in its most recent generation), and a 4GL programming language for rapid application prototyping. It was used to collect the first SWITCHBOARD corpus [1], and was also used on the CALL HOME and Voice Across Hispanic America (VAHA) projects [2].

What was wrong with this system? First and foremost, it is a closed architecture based on a platform that is incompatible with those most commonly used in speech research. Hence, its acceptance by the community as a general purpose platform has been slow. Second, for most projects extensive firmware modifications have been required by the vendor to perform a particular style of data collection. Such modifications have historically been very expensive and have caused numerous delays to the projects requiring them. Third, the current platform requires a steep learning curve involving a nontrivial custom environment. Operators often have to be educated directly by the vendor and require several months to come up to speed. The cost of maintaining and operating the system is extremely high. Lastly, the proprietary interfaces require designers to completely rewrite all applications when moving to new hardware.

Platform independence was a major design goal for this project — so why not Java? Sun's Java webpage describes the Java Speech API, which will allow Java applications and applets to incorporate speech technology into user interfaces. It will leverage the audio capabilities of other Java Media APIs, and when combined with the Java Telephony API, will support advanced computer telephony integration [3]. The first release of this API occured in October 1998, after the completion of the project.

The system described in this paper is tailor-made for collecting data for speech processing research. The platform uses a very inexpensive and popular workstation, a Sun Sparcstation 5, as its host. The hardware vendor's general-purpose API [4] has been encapsulated in C++ to abstract the hardware. The end user exploits a graphical user interface to easily build complex applications. The high-level data collection software (including the GUI) is hardware independent, making the transition to other hardware systems viable.

HARDWARE

T1 interfaces for Sun workstations are hard to find. The market leader in this niche arena is a system developed by Linkon Corporation. This system includes a Newbridge T1 communications card that occupies one SBus slot and performs all data transmission functions, and a two-slot multi-DSP module that handles all call processing. The Linkon FS4000 board [4] is capable of independently recording both sides of a bridged conversation synchronously, a vital feature for current collection efforts. Linkon packages their board with some low-level, general-purpose software which serves as the foundation for building telecommunication applications.

SOFTWARE

The Linkon hardware was not specifically designed with speech data collection in mind, but rather to be a general purpose telecommunications board. As illustrated in Figure 1, the code developed for the system is built as a multi-level hierarchical C++ framework, which can be accessed from many different levels. The *Data_collect*, *Item*, and *Item_list* classes are highly interdependent; they are meant to be driven by the GUI and parameter file interface rather than hard-coded settings. The *Tele_interf* class interfaces with the hardware, allowing a wide variety of applications in C++ without dealing directly with any device drivers. Below the *Tele_interf* class resides a collection of low-level classes intent on performing speech related tasks through the general-purpose hardware API. In order to support a new hardware platform, the system designer need only revise the *Tele_interf* class and all other software and applications will follow.

All software above *Tele_interf* directly supports the development of parameter-file driven speech data collection applications. *Item*s are self-executing objects which are runtime-configurable to perform specific tasks. The call-flow of an application is completely described through a list of properly configured *Item*s, as shown in Figure 2. The *Data_collect* class performs the required bookkeeping for a serious data collection effort, including directory organization, call status logging, and the auxiliary exception notification necessary for a dedicated system.

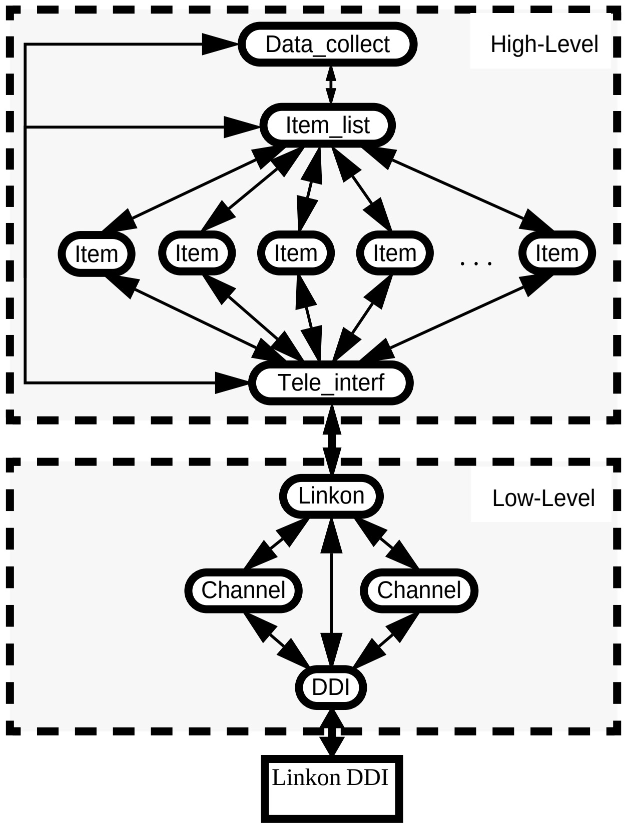


Figure 1. Software Hierarchy

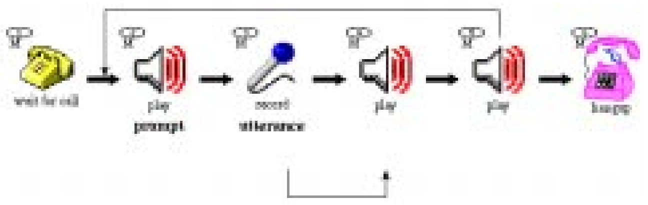


Figure 2. Call-Flow Represented as a Collection of Items

One vast improvement in this system to general purpose telephony hardware is that audio files are stored in a format accepted by the speech recognition community, eliminating part of the post-processing necessary for releasing a speech corpus. This system directly uses the NIST SPHERE audio file format [5] for both predefined dialog prompts and recorded data. This non-proprietary format is the standard for distributing speech corpora. Database information is stored as a header in the same file as the audio data, greatly simplifying the necessary bookkeeping. In addition, a custom endpointer [6] is available to automatically crop recorded audio to boundaries which maximize the performance of state-of-the-art speech recognition technology.

GRAPHICAL USER INTERFACE

The easiest way to rapidly prototype a telephone speech data collection application is to use the graphically based application builder. This tool is the highest level interface available for the system. It allows the user access to all possible parameters in all situations, thus providing a simple and consistent interface to application development. On-line, context-sensitive help is always available, detailing the specifics of each parameter for the given *Item* type. A screen capture of the application builder is shown in Figure 3.

The application is displayed in two formats simultaneously. The top half of the screen shows the call flow graphically; icons represent the call flow items and arrows represent the control and data flows. The bottom of the screen uses text-based lists to show the call flow items and key parameter settings. This multi-modal interface is optimized to have an easy learning curve for novice users while not encumbering experienced users. Multiple sets of naive users assisted in human factors testing to improve the tool's ease-of-use.

The graphical application debug mode maps a running application to the screen in real time. To use this functionality, the driver application is run in a special mode which outputs intermediate debug information. The application builder tool can then poll this debug information and automatically update the display in response to the running system. The current item selector box moves across the screen as the driver application moves through the call flow. The key internal data (such as user key presses, database query results, or speech recognition hypotheses) is displayed on the screen. This debugging facility completes the suite of tools necessary for real application development.

SWB COLLECTION

The Linguistic Data Consortium's SWITCHBOARD-2 (LDC SWB-2) protocol defines a very sophisticated data collection system [7]. Many issues went into the application's design, some dating back to simpler data collection efforts years ago at Texas Instruments. A few dialog exchanges fulfill legal requirements (such as the 'press 1 to participate'), while others assist in correcting physical shortcomings in the phone network (such as unreliable ANI information).

To collect a SWITCHBOARD database, a large number of volunteers must register into a database, providing a list of availability times and phone numbers. The participants then call into the system at their leisure, using a PIN to identify themselves. A second caller is located by the system using the database of available participants, up to 5 calls will be attempted. Once both callers are on-line, a topic prompt will be read and the conversational speech recorded. The database is designed such that two participants will never talk to each other twice. Also, each call must be initiated from a unique phone number, as gathered through the RT-ANI packets (commercially known as caller-id) from the digital phone network.

Phase 3 of the SWB-2 corpus was gathered from Southeastern universities during the fourth quarter of 1997. To test the versatility of our system in large applications, the SWB-2 protocol was fully implemented to mimic LDC's Intervoice system. All database methods remotely query the LDC Oracle server, utilizing the primary speaker database through custom Perl network communication scripts. This pilot application is currently being refined by LDC staff for use in future collections.

CONCLUSIONS

We have developed a robust, fully-expandable system for platform-independent collection of telephone speech data. Our object-oriented software libraries and easy-to-use GUI provide powerful tools with which even a novice user can efficiently create complex applications. While the current focus of the speech research community is the recognition of broadcast news (LDC's HUB-4), significant effort is still expended towards SWITCHBOARD-type telephone data, for which this system is poised to play a pivotal role.

All software and documentation for this project is available via the Institute for Signal and Information Processing's web page [8].

REFERENCES

[1] J.J. Godfrey, E.C. Holliman and J. McDaniel, "SWITCHBOARD: Telephone Speech Corpus for Research and Development," *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. I-517-I-520, San Francisco, California, USA, March 1992.

[2] B. Wheatley and J. Picone, "Voice Across America: Toward Robust Speaker Independent Speech Recognition For Telecommunications Applications," *Digital Signal Processing: A Review Journal*, vol. 1, no. 2, pp. 45-64, April 1991.

[3] "Java Speech API: A White Paper," *http://java.sun.com/products/java-media/speech/*, Sun Microsystems, 1998.

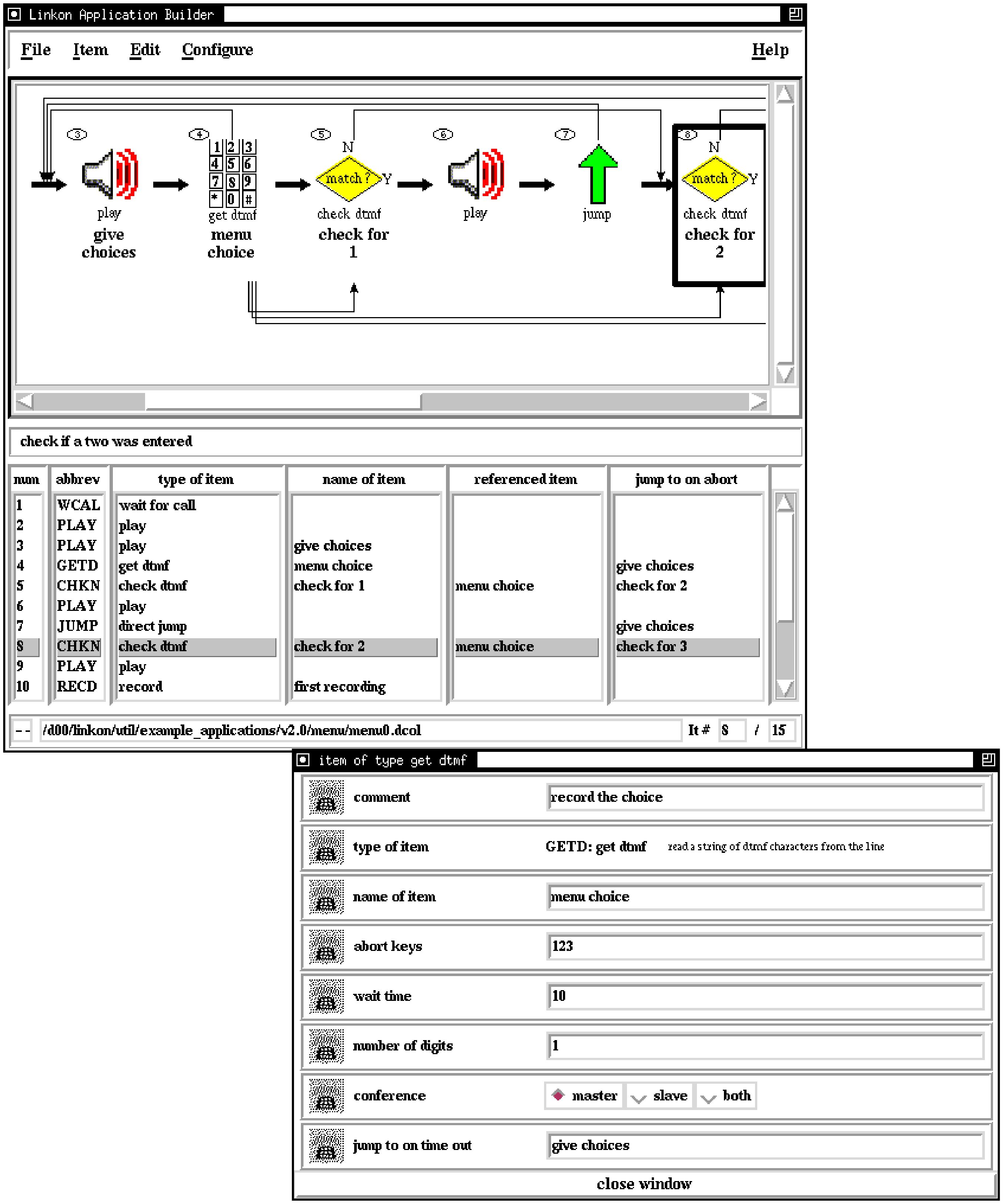
[4] Linkon Communications Board Direct Driver Interface Programmer's Guide and Reference Manual, rel. 5.0.0, Linkon Corporation, July 1996.

[5] J. Fiscus and J. Garafolo, "Speech Header Resources (SPHERE)", *http://www.itl.nist.gov/div894/894.01/software.htm*, National Institute for Standards and Technology, May 1993.

[6] A. Ganapathiraju and J. Picone, "Echo Cancellation For Evaluating Speaker Identification Technology," *Proceedings of IEEE Southeastcon*, pp. 100-102, Blacksburg, Virginia, USA, April 1997.

[7] "Switchboard-2 Phase 1", *http://morph.ldc.upenn.edu/Catalog/LDC98S75.html*, Linguistic Data Consortium, 1998.

[8] R. Duncan and J. Picone, "The Linkon Telephone Data Collection System Users Guide," *http://www.isip.msstate.edu/projects/t1_interface/*, Mississippi State University, March 1998.



check if a two was entered

num	abbrev	type of item	name of item	referenced item	jump to on abort
1	WCAL	wait for call			
2	PLAY	play			
3	PLAY	play	give choices		
4	GETD	get dtmf	menu choice		give choices
5	CHKN	check dtmf	check for 1	menu choice	check for 2
6	PLAY	play			
7	JUMP	direct jump			give choices
8	CHKN	check dtmf	check for 2	menu choice	check for 3
9	PLAY	play			
10	RECD	record	first recording		

-- /d00/linkon/util/example_applications/v2.0/menu/menu0.dcol It # 8 / 15

comment	record the choice
type of item	GETD: get dtmf read a string of dtmf characters from the line
name of item	menu choice
abort keys	123
wait time	10
number of digits	1
conference	[✓ master] slave both
jump to on time out	give choices

Figure 3. The Application Builder Designs a Prompt-Record Style Data Collection Application