WAVEEDIT, AN INTERACTIVE SPEECH PROCESSING ENVIRONMENT FOR MICROSOFT WINDOWS PLATFORM

M. Akbar

Laboratoire de la Communication Langagière Interaction Personne Système
Université de Joseph Fourier, 38041 Grenoble cedex 9, France
Tel. +33 4 76 51 45 26, FAX: +33 4 76 44 66 75, Email: Mohammad.Akbar@imag.fr

ABSTRACT

This paper presents a new interactive speech processing environment designed for Microsoft Window platforms. It will be shown that how an integrated speech processing environment was made following Windows Interface Design Guidelines. The environment integrates many traditional time and frequency domain analysis algorithms as well as basic functions like recording, listening and labeling. Choosing Component Object Model (COM) as the architectural framework assures high maintainability, scripting capability and further expandability of this environment. Extensive use of the system in laboratory has shown how this interactive environment improves users performance in their every day speech processing tasks.

1 INTRODUCTION

Today, personal computers offer such a powerful processing performance that running big scientific applications on PCs becomes a simple task. Users' working-model is rapidly changing and they constantly need more interactive, sophisticated and flexible environments in their work. A number of signal processing packages are available even in PC platform, but in speech processing we specially work with huge amount of contiguous data and these packages are not really adapted for such tasks. We need to synchronize long streams of data, browse in and compare them rapidly. We need to label signals automatically and/or manually in many distinct levels of detail and easily modify them when needed.

For many years Unix stations offered reliable and maintainable solutions where one could have sufficient continuity and compatibility between different versions of a product. But the Unix based solutions are still quite expensive. The complex nature of interactivity, working with huge amount of data and the cost performance ratio leaded us to develop such an interactive environment on Microsoft Windows 32 bit platforms (Windows NT and Window 95). This selection provides an attractive environment for every day tasks of speech processing while proposing much less expensive solutions than the comparable Unix based solutions.

In the other hand to keep in pace with ever changing world of computing we had to choose an architectural framework that would allow good modularity and maintainability of our system. To have a longer system's life cycle we needed a component based framework in which we integrated a scripting language that allows end users to further expand their working environment by adding new components. This component approach allows us to dynamically replace or add elements without side effects on the other parts.

This paper presents the implemented environment, and shows how COM architectural framework helped to develop an open and easily configurable package. Section 2 presents some architectural details of current system and section 3 describes WaveEdit user interface and signal processing features in more detail. We conclude our description by presenting some real situations in which WaveEdit could be used.

2 WAVEEDIT ARCHITECTURE

Since many years object oriented programming (OOP) paradigm has helped many system designers to create easily maintainable environments. OOP like many other paradigms was subject to some important evolutions. Today we talk about self contained components in OOP. While traditional objects follow a hierarchy to implement the same interfaces, components support one or more *abstract* interfaces out of hierarchy. By supporting a specific interface a component provides the same functionality on his specific data type. The key is that components don't need to keep a heritage relationship to provide the same operators on their specific data types. Once a component implements an interface (each interface is known by its Global Unique ID number) it can called regardless of the data type it supports.

However there are many important points about software engineering side of the component paradigm (COM) [2], our goal will be to show how components build up WaveEdit interactive environment and how the environment could be used in some real projects.

WaveEdit is thus composed of some interconnected components providing system functions. The major components of current WaveEdit package are shown in hierarchy order in Fig. 1. Each component has a predefined interface and other objects have access to object methods only through these interfaces. Some generic interfaces have been also defined to simplify addition of new components.

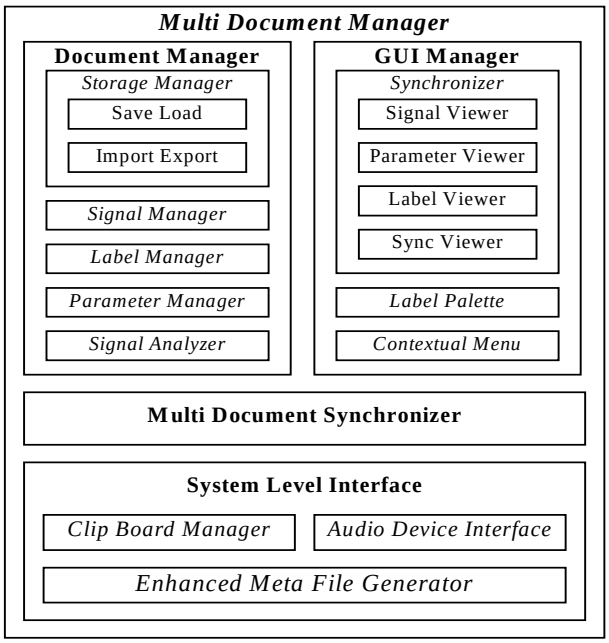

Fig. 1. WaveEdit environment major components hierarchy

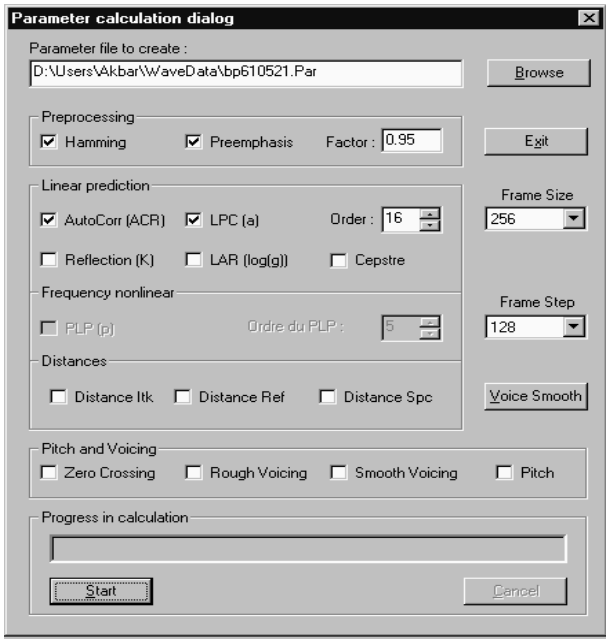

Fig. 2 WaveEdit Signal Analyzer main dialog

As it is obvious from Fig. 1 WaveEdit is largely made of a *Multi Document Manager (MDM)* that allows working on multiple signals and parameter sets at a time. Different documents may be synchronized by messages passed through *MDM*. It can thus contain more than one *Document Manager* (DM) and *Graphical User Interface Manager* (GUIM) at a time. In WaveEdit terms a document is a combination of a sound file, it's associated parameters (acoustic, spectral,…) and labels. Some other configuration variables are also associated to each document in order to keep user preferences. Each document has a graphical representation at run time that is essentially made of four components: *Signal Viewer, Parameter Viewer, Label Viewer, and Sync Viewer*. The first three components are simply time representations of their respective information. They provide scaling, translation and selection on these primitives. The fourth one is a separate representation providing a cross section of spectral or some other suitable parameter arrays synchronized by a label's position. *GUIM* provides also some other important components like a dynamic *label palette* --user can choose a label from palette and place it in the documents-- and a *contextual menu* --this gives a fast access to the mostly used commands.

As said before, each document keeps track of three connected components: Signal, Labels and Parameters. *DM* provides methods to *save* and *load* them in a compact native format. At the same time it provides *import* and *export* facilities to convert between major existing data types and native format. *DM* is supported by *Signal, Label, and Parameter Managers* to provide basic operations on these data types. Their most important function is to provide fast access to very big resources. This has been achieved by clever memory management and optimization. As a example CD quality signals of even 2 hours long, a 660 MB file, could be easily worked on. *Signal Manager* also provides *recorder/player* interface directly used by *Signal Viewer* to provide integrated recording/listening, facilities to the system.

Document Manager is composed of another major internal component called *Signal Analyzer*. As it's name stands for, this component answers to the signal processing needs of the system. *Parameter Manager* uses the *Signal Analyzer* to calculate parameters of input raw signal (Fig. 2) provided by *Signal Manager*. Calculated parameters are stored in a file and can be accessed through *Parameter Manager* Interface. In some cases *Parameter Viewer* may directly use the same component to calculate parameters on the fly. This approach trades memory and disk usage by slightly increasing processing time. The major parameter array calculated this way is a running spectrogram that if implemented by *Parameter Manager* could drastically increase disk space usage. The *Signal Analyzer* is itself composed of some low-level components that provide different type of signal processing. These include spectral (e.g. LPC, LAR, PLP, and Ear Model), voiced/unvoiced, pitch, and some acoustical parameters directly derived from production models.

Although not shown in Fig. 1, there are some other components that create the real user interface of system, provide cut, copy, paste facility, and do low level signal processing, recording, playing and so on. In the next section while presenting the actual system you may see the major components from which the WaveEdit system is made up. As an interesting example Fig. 4 shows a graphical representation of a document directly inserted into a Word document using *MetaFile Generator* component.

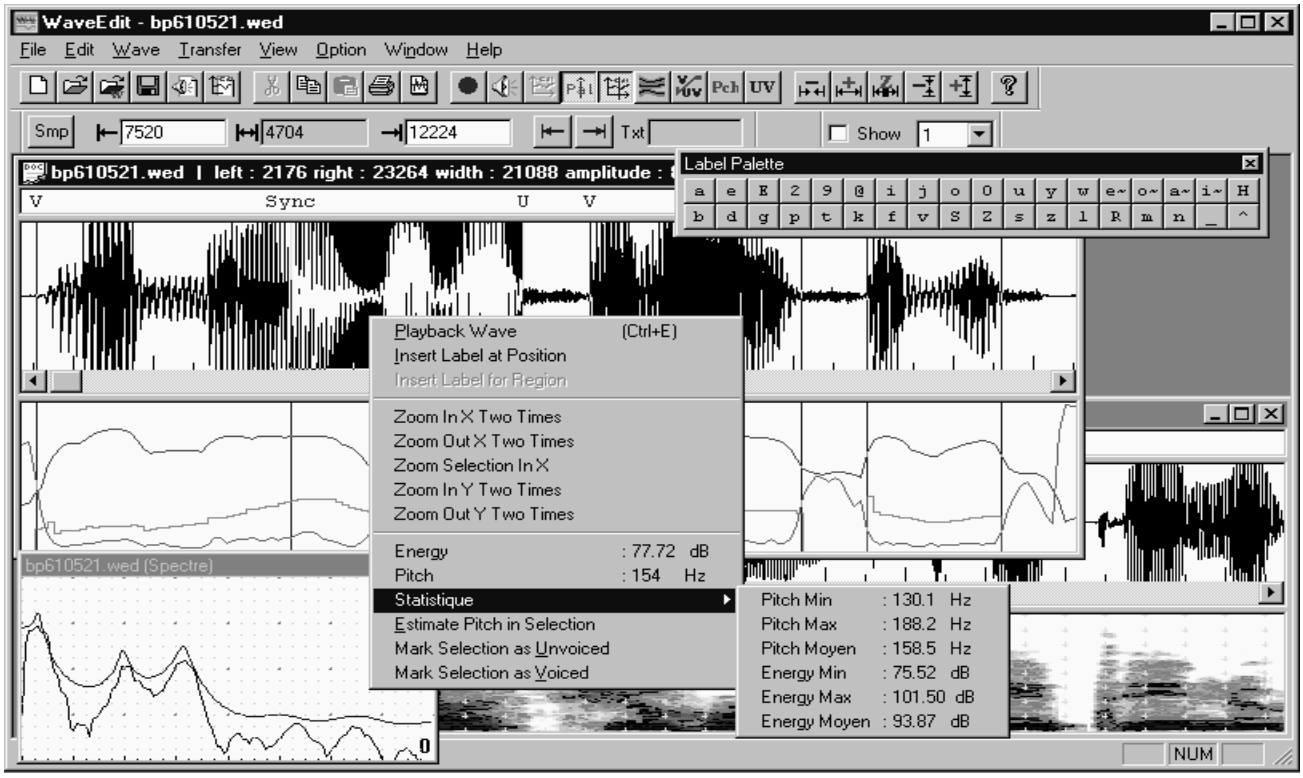

Fig. 3 A general look of WaveEdit environment in use

3 ACTUAL WAVEEDIT SYSTEM

By the time the current version of WaveEdit Environment is 1.05 beta and entirely implemented on Visual C++. However any other programming language supporting COM (e.g. Microsoft Visual Basic, Borland C++) may be used to create and add new features to the environment.

In a user interface point of view respecting Windows 95 Interface Guidelines [6] allowed us to design a harmonized and easily understandable interface for Windows 95 and Windows NT 4.0 users.

In a signal processing point of view many traditional algorithms regarding prosodic studies have been implemented in the current system. These include a proprietary Voicing Analysis based on Neural Network classifiers that uses spectral parameters to determine voicing state of speech signals. This NN comes pre-trained as a help to the user to bootstrap his data set but users may ask for online retraining based on a labeled speech signal (or corpus) to improve NN performance for specific data. A fast and accurate pitch-tracking component, based on a modified AMDF evaluation, has been also added to Signal Analyzer component.

Current WaveEdit environment has been used in some projects in the laboratory. These projects were essentially about manually and automatically labeling of large speech corpora. As an example two, CD-ROMs including more than 15 hours of continuous speech have been produced and labeled entirely using the WaveEdit environment for AUPELF-UREF ARC B2 [8] task. These CD-ROMs are available through the aforementioned action. In another project a large speech corpus created in order to study relations between speech act and prosodic features has been entirely labeled [9]. The produced corpus is provided on a single CD-ROM and will be available in a near future.

To better understand the features of current implementation a brief description of major functions of WaveEdit environment is presented in the following.

3.1 Visual Interface Features

Fig. 3 shows a general look of WaveEdit 1.05 environment in use. Its Major visual interface features are listed below:

- Multiple time synchronized documents,
- Multiple view on a single documents containing Signal, Parameters and Labels,
- Synchronized view of document elements,
- Selective playback of regions,
- Multiple level selective label editing,
- Scaling functions (in time and amplitude),
- Contextual menus allowing rapid access to available commands in each editing region,
- User configurable dockable/floating label palette,
- Dockable or floating command palettes for fast access to the frequently used commands,
- A large set of customization options,
- Copy/Paste to other documents,

3.2 Signal Processing Features

WaveEdit 1.05 implements the following list of well-known signal processing algorithms and related functions:

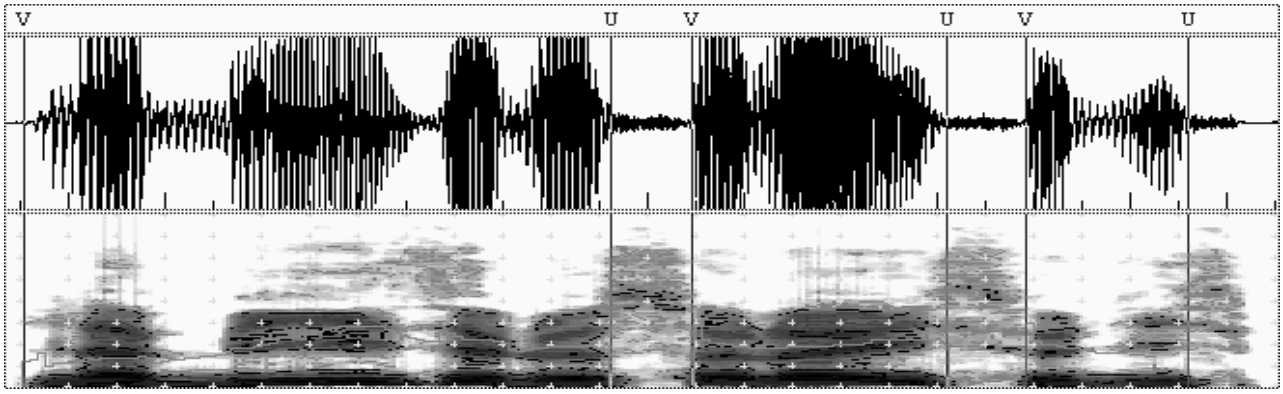

Fig. 4. An example of inserted WaveEdit document into Word

- Recording functions (selective sampling frequency and format with optional real time silence detection feature useful in recording large corpora),
- Energy, Zero crossing, Pitch frequency measures,
- Autocorrelation, Linear Prediction [4], Log Area Ratios[4], Reflection[4], Cepstral [4] and Ear Model[7] parameters calculation,
- Spectral distance measures,
- Statistical measures on Energy and Pitch including Mean, Variance, Min, Max,
- Real time Spectrogram calculation with adjustable band width and temporal resolution through user interface,
- Cursor synchronized Spectrum, Autocorrelation, AMDF,… calculation,
- Proprietary pitch tracking algorithm,
- Rupture model derived segmentation [1],
- Neural Network derived segmentation [5],
- Vector Quantization derived segmentation [3].

As it's stated before improvements and additions to the current list is easily provided by writing new algorithms, following the component writing style and adding them to the list of supported components.

3.3 Data Base and File Processing Features

In WaveEdit 1.05 keeping track of large corpora is made possible by using a tree structure corpus architecture that indexes on recorded signal, spoken text, associated label information, speaker information, repetition count and recording conditions (date, audio recording material used,…). In the following we present a part of functions supported in this category:

- Unlimited input file length,
- Tree structure corpus creation/modification,
- Multiple open documents at a time (MDI),
- Direct support of PCM 8 and 16 bits Wave and Raw format by Signal Manager,
- Support of other Wave formats (e.g. DPCM, ADPCM) through Import/Export tool,
- Proprietary parameter and label files format, convertible to human readable text files (TIMIT, HTK and PTS formats are supported),
- Printing and Exporting Image of selected regions.

4 CONCLUSIONS

In this paper a highly interactive speech-processing environment specially under Windows 95 and Windows NT platforms has been presented. The choice of platform and architectural framework has been explained. These choices leaded to a low cost solution for speech processing. The features of current implementation were briefly described. We have also presented some real situations in which WaveEdit has been used successfully. Those applications include segmentation applications and corpus generation tasks.

5 REFERENCES

[1] R. André-Obrecht., "A new statistical approach for the automatic segmentation of continuous speech signals", *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, no. 1, pp. 29-40, January 1988.

[2] D. Rogerson, *Inside COM Microsoft's Component Object Model*, Microsoft Press, 1997.

[3] J. Makhoul, S. Roucos, H. Gish, "Vector Quantization in Speech Coding", *Proceedings of the IEEE*, vol. 73, no. 11, 1551-1588, November 1985.

[4] J. Makhoul, "Linear Prediction: A Tutorial Review", *Proceedings of the IEEE*, vol. 63, no. 4, 561-580, April 1985.

[5] Richard P. Lippman, "Review of Neural networks for Speech recognition", *Neural Communication 1*, pp. 1-38, 1989.

[6] N.W. Cluts, *Programming the Windows 95 User Interface*, Microsoft Corporation, 1995.

[7] J. Caelen, M.K. Nasri, E. Reynier, H. Tattegrain, "Architecture et fonctionnement du système DIRA. De l'acoustique aux niveaux linguistiques", *Traitement du Signal*, vol. 7, no. 4, 1990.

[8] AUPELF-UREF, Action de recherche concertée, "Linguistique, Informatique et Corpus Oraux: Dialogue Oral", CD-ROM OTG 1 & 2, 1997.

[9] CAELEN-HAUMONT, G. et BESSAC, M. La prosodie, de la sémantique à la pragmatique. In : RSP, *Actes des Rencontres de Sémantique et Pragmatique*, Saint-Denis, 4-5 octobre, 1996. Paris : Université Paris III, 1996.