Mapaci: a Real Time e-Health Application to Assist Throat Complaint Patients

Jesus Bobadilla¹, Pedro Gomez², Juan I. Godino²

¹IA, Escuela Universitaria de Informatica, U.P.M. Complejo Politecnico de Vallecas
Km.7 de la Carretera de Valencia, 28031, Madrid Spain
jbobi@eui.upm.es

²GIAPSI, Facultad de Informatica, U.P.M. Campus de Montegancedo s/n 28660, Boadilla del Monte (Madrid) Spain
pedro@pino.datsi.fi.upm.es

Abstract

This paper describes the fundamentals of a complete e-Health Web cooperative information system combining the following features: a research and development platform for telemedicine and patient screening including new voice processing algorithms and a communications framework. The main goals and design decisions, related to architectural analysis of this information system oriented to voice pathology are pointed out. A methodology to handle patients’ voice samples, enabling real-time remote capture, transmission, storage and processing is documented. These features allow to define a complete information system including database recruiting as well as cooperative diagnose. Hints on the use of the Java Media Framework© (JMF) technology and the use of different protocols to implement synchronous services, including audioconferencing and videoconferencing are documented. The preliminary results show the feasibility of the design. Expected benefits are among others an improvement in the knowledge of voice pathology evolution, and the remote screening of large population sectors.

Keywords: e-health, speech, JMF, protocol, real-time

1. Introduction

Nowadays, it is possible and convenient to promote the use of e-health applications in order to obtain both medical and technical research improvements. This paper shows the main goals and design details of the MAPACI¹ e-health project.

Actually there exists a large number of web based e-health services where people can obtain different types of medical information [1],[2],[3]. Additionally, we can find e-health projects where Internet is used to obtain and process some types of medical data (speech, analysis results, radiographies…) [4],[5],[6]

MAPACI offers a complete system where medical specialists can use remote access to a speech capturer ad-hoc program, record on-line samples of patients with some type of problems in their throats, send the speech to a host and, finally, record the samples in a standard format. Each medical speech file is linked to a set of different type of technical and personal record information of its corresponding patient.

The different records obtained all over the national medical network are stored both into a database and in a web site as a set of well structured web pages prepared to be crawled using specialized web data mining multiagent spiders [7],[8].

MAPACI project develop specialized signal processing algorithms [9],[10] to obtain mathematical models of biological parameters. The numerical results of the signal processing algorithms are processed using neural networks [11],[12] in order to obtain automatic diagnostics that can be tuned which each new speech record that arrives to the server host.

In this paper we will present the architectural design and the technological aspects of the MAPACI project, rather than the signal processing or artificial intelligence ones.

To develop the required application we must deal with two mayor problems: the first one is to capture the speech and access to the individual samples, and the second one to deal with the Java virtual machine [13] security manager restrictions both in the capture and the transmission stages. We must keep in mind that one of our goals is to provide a secure and easy way for the clients (mainly doctors) to access to our application latest versions over the time. Java applets [14] offer this possibility, as well as the multiplatform support.
2. Application definition

This project tries to benefit both the patients with some physical speech medical problem and the community of medical and engineer researches working in this area.

People usually visit their family doctor for any type of medical problem. In some cases, family doctors can manage the situation, but more complicated ills need the attention of specialized doctors using more sophisticated technical material.

The use of MAPACI begins into the family doctor consulting room, when a patient presents a physical speech problem and the family doctor decides to consult to the corresponding specialized doctors. In this case, MAPACI offers a direct and immediate communication support to a pool of otorhinolaryngologist which can help the patient using synchronous services: chat, talk, RTP audioconference, RTP videoconference.

In spite of these difficulties, there are different applications available, which were originally not professional and have been created from scratch in academic environments [15],[16],[17],[18]. Usually, these applications require continuous maintenance to offer the latest coding standards and to admit the different audio cards that come out on the market.

To aid the development of multimedia applications, Sun Microsystems© provides Java Media Framework (JMF) API [19]. The most evident advantage of its use is the possibility to access the video and audio signal through a wide group of classes and interfaces; a less immediate but very important advantage is that we can rely on Sun Microsystems to provide support for new codecs and audio cards that come out on the market.

Unfortunately, using the JMF API appropriately is not at all simple. The documentation that exists is based on the document JMF API Guide [20] and the few books published on the subject [21][22], in which it does not specify how to access the individual samples, or the detailed way to create effects. The rest is an open search of the Internet, where at most we can find examples of the general use of streams, without reaching the level of samples processing.

The JMF architecture has four levels (Fig. 2): user applications used by JMF API are developed at the top level; the second level offers objects that enable real-time capturing and visualizing of different types of data flow (streams), likewise it provides the capacity of carrying out standard processing on these streams.

Using MAPACI is possible too to record speech samples and send them to the server host, where a database will be used to store web based medical records forms. The pool of specialists has immediate access to the database.

Finally, MAPACI offers useful signal processing algorithms which present different real time results to be used for the pool of otorhinolaryngologists.

The collected database will benefit researchers both in the medical and engineer areas, making possible to improve the medical diagnostics as well as the engineer results.

3. Capturing the speech

Researchers encounter technical difficulties accessing the samples captured by the audio cards due to the following: the type of hardware used, lack of technical documentation on them, protection of the operating systems in low-level access, different coding and compression of the audio signal, updating drivers, etc., which makes the development of non-professional applications for real-time signal processing much more difficult.

Unfortunately, using the JMF API appropriately is not at all simple. The documentation that exists is based on the document JMF API Guide [20] and the few books published on the subject [21][22], in which it does not specify how to access the individual samples, or the detailed way to create effects. The rest is an open search of the Internet, where at most we can find examples of the general use of streams, without reaching the level of samples processing.

The JMF architecture has four levels (Fig. 2): user applications used by JMF API are developed at the top level; the second level offers objects that enable real-time capturing and visualizing of different types of data flow (streams), likewise it provides the capacity of carrying out standard processing on these streams.

Unfortunately, using the JMF API appropriately is not at all simple. The documentation that exists is based on the document JMF API Guide [20] and the few books published on the subject [21][22], in which it does not specify how to access the individual samples, or the detailed way to create effects. The rest is an open search of the Internet, where at most we can find examples of the general use of streams, without reaching the level of samples processing.

The JMF architecture has four levels (Fig. 2): user applications used by JMF API are developed at the top level; the second level offers objects that enable real-time capturing and visualizing of different types of data flow (streams), likewise it provides the capacity of carrying out standard processing on these streams.

Researchers encounter technical difficulties accessing the samples captured by the audio cards due to the following: the type of hardware used, lack of technical documentation on them, protection of the operating systems in low-level access, different coding and compression of the audio signal, updating drivers, etc., which makes the development of non-professional applications for real-time signal processing much more difficult.

Unfortunately, using the JMF API appropriately is not at all simple. The documentation that exists is based on the document JMF API Guide [20] and the few books published on the subject [21][22], in which it does not specify how to access the individual samples, or the detailed way to create effects. The rest is an open search of the Internet, where at most we can find examples of the general use of streams, without reaching the level of samples processing.

The JMF architecture has four levels (Fig. 2): user applications used by JMF API are developed at the top level; the second level offers objects that enable real-time capturing and visualizing of different types of data flow (streams), likewise it provides the capacity of carrying out standard processing on these streams.
4. Accessing to the individual samples

To access individual samples, JMF provides the possibility of implement an effect plug-in, (Effect is a JMF interface), as you can see in figure 1. The Effect interface can be implemented as a personalized AudioEffectFramework class (figure 3), that will support the most important processing of our real time application.

Figure 3 shows the proposed schema of a typical application which manages individual speech samples:

![RealTimeApplication](image1.png)

**Fig. 3. Application architecture**

At the highest level the RealTimeApplication uses the presentation levels and JMF plug-in API. The AudioEffectFramework class implements an effect that we will define in the AudioEffect class. Finally, the IOFormats class defines the input and output formats admitted by the effect.

The application captures an audio stream with input format defined in IOFormats, it can process the samples and converts it to output format, also defined in IOFormats; finally, it sends (or renders) it to an output device (Fig. 4).

For each specific number of samples that the input device gives us it executes a method `process` of the AudioEffectFramework class (Fig. 5). This method invokes the implemented algorithm that we can implement in the AudioEffect class.

![RealTimeApplication](image2.png)

**Fig. 4. Stream processing**

5. Sending information to the host

The natural way to send (to the host server) the captured speech data would be to use the real time protocol (RTP) [23] and transmit this data in real time; it can be programmed using JMF [20]. Nevertheless, due to the Internet variable wideband, we could loss fractions of the data along the communication process. MAPACI uses RTP to implement audio and videoconference, but not to send and record all the patient speech samples.

When it is necessary to save all the speech samples into the server, RTP is not the appropriated protocol. FTP would be the best option in a typical application [24], but MAPACI is not implemented as a Java application, instead of it MAPACI is a Java applet. Java applets offer different advantages, mainly the immediate and safe distribution code via Internet.

Using java applets we pay a price: the security Java virtual machine (JVM) restrictions [13]. For security reasons we can not save a file in the client computer and we can only capture samples if the client has installed JMF.

Because it is not possible to save information into the client computer file system, MAPACI save all the samples and some technical parameters (sampling frequency, number of samples, bits per sample, etc.) into the client computer memory, and later it sends all the data to the host using the TCP/IP protocol [25]. When all the data arrives to the host, MAPACI saves this data in WAVE format.

Figure 6 presents a table showing the different protocols used into the MAPACI application:

<table>
<thead>
<tr>
<th>Service</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audioconference, videoconference</td>
<td>Real Time Protocol (RTP)</td>
</tr>
<tr>
<td>Chat, talk</td>
<td>User Datagram Protocol (UDP)</td>
</tr>
<tr>
<td>Recorded speech transfer</td>
<td>Transmission Control Protocol / Internet protocol (TCP/IP)</td>
</tr>
<tr>
<td>Patients personal information</td>
<td>Hyper Text Transfer Protocol (HTTP)</td>
</tr>
</tbody>
</table>

6. Application architecture

MAPACI project software architecture is not simple, because the project itself covers a wide range of goals and possibilities. Figure 7 shows a very simplified diagram with the more important objects of the complete design.
Using part of the MAPACI objects we have developed an API and we have written the associated technical information; all this will be published very soon as: “development of applications for real-time audio and video signal processing”.

MAPACI software, at the lowest level, is supported by Java Development Kit (JDK) and Java Media Framework, using both their presentation/processing classes and the RTP classes. At the upper level, we separate the client side and the server side. Finally, the signal processing algorithms and the visualization methods form the research kernel of the project. The mathematical details of them will be published in different signal processing papers.

![MAPACI Architecture](image)

**Fig. 7. MAPACI architecture**

The objects situated at the left side of figure 7 implement the speech capture and send to host capabilities of MAPACI. The centred classes implement the audio and video RTP conference, and finally, the right side objects support the UDP chat and talk services.

7. Results

MAPACI software is actually running as a prototype. Its use depends on the interest of the group of family doctors involved in the implantation of the system, as well as the otorhinolaryngologists team work.

The application user interface has been simplified as far as possible in order to present a friendly aspect to the doctors. All the MAPACI services actually run in perfect real time, except of course, the videoconference, which depends on the available wideband at each session (we do not use dedicated channels with fixed bitrate).

Because we are accessing to the speech individual samples, it is possible to implement signal processing algorithms and generate real time results, even in the client side; for instance, the patient would try to modulate his speech depending on the real time plots MAPACI presents. To study this possibility, we took a compressed speech signal applying different processing loads, and then we evaluated the impact on the speech bitrate (figure 8). The results show it exist a significant quantity of signal processing algorithms we can apply in real time.

![Performance Results](image)

**Fig. 8. Performance results**

8. Conclusions

MAPACI project is a complete and useful ehealth application which offers and combines research and development, speech signal processing algorithms and communications, society (patients) benefits, medicine’s tool and engineer’s platform.

Even with just the application prototype implanted, it is possible to offer a satisfactory solution in the otorhinolaryngologists e-health area. Besides this, actually MAPACI acts as a platform where different state of the art speech signal algorithms are implemented.
The database containing the incoming speech and personal records of the patients will be used to improve both the human and computer generated diagnostics, and will contribute to reach medical and engineer advances in general and to benefit patients in particular (we hope so and we will work for it).

9. References

[5] tie.telemed.org/programs
[6] www.isis.georgetown.edu