MULTIPAS: JAVA, C++ AND C# TO OCTAVE BRIDGES

Begoña García, Amaia Méndez, Ibon Ruiz, Javier Vicente
mbgarcia@eside.deusto.es, amendez@eside.deusto.es, ibruiz@eside.deusto.es, jvicente@eside.deusto.es

ESIDE, University of Deusto. Bilbao. Spain

ABSTRACT

This proposal introduces the APIs developed by PAS research group from the University of Deusto (UD) to signal processing learners three gateways between the main programming languages (Java, C++ and C#) and "Octave" mathematical software. These APIS enable the use of "Octave" variables and functions through a JAVA- C++- C# program and makes it possible to not only develop signal processing applications quickly by implementing the application's graphic interfaces but also to carry out the scientific calculation in "Octave". These gateways are developed with the shape of API, and have been made available to students doing degrees in Electronics and Telecommunications Engineering, so as to assist them in their lab training in signal processing, as well as for the drafting of their final projects. . They are very useful to research developments as well, not to rewrite existing code. Finally, it is important to stress the free-software nature of the developed gateways: as no license is necessary, student access to this program of scientific calculus is easier.

Index Terms— *Engineering Education, API, C++, C#, Java, Octave*

1. INTRODUCTION

The development of the "joPAS – coPAS - monoPAS" programming APIs arose from the PAS research group's need to develop prototypes of the algorithms used with a user's interface in biomedical signal processing research area. Several of our algorithms were developed in "Octave" language, which, despite being ideal for signal processing algorithms, lacks a user interface. Therefore, so as to program a demo application of the work carried out by the team, we saw it necessary to re-implement the code in another programming language that enables the programming of graphic interfaces. This meant that the team would lose out on research performance in order to devote more time to less productive tasks. Thus the need for a tool allowing the reuse of "Octave" developed algorithms as well as providing the possibility of user interfaces.

Once the APIs had been developed, their great potential had not only for research work and application development, but also for the sphere of education, due to its user friendliness.

At present, students of both electronic and telecommunications engineering (in University of Deusto) have plentiful knowledge of digital signal processing, as it is one of the fundamental pillars of their training. During their degree or Ph.D studies they learn how to understand and develop mathematical algorithms in order to characterize voice behavior, image sequences or control systems. Concretely, they use MultiPAS to implement a research project about pathological vocal folds segmentation algorithm and oesophageal voice improvement system.

2. OBJECTIVES

The main aim of the APIs application is to make a tool available that can enable the rapid development of applications with a user interface that use signal processing algorithms implemented with other tools (Octave). This being the main aim, a number of secondary objectives could also be achieved:

- ❖ To reduce the cost of licences for mathematical programs, through the use of free software, such as Octave, which specialises in digital signal processing.
- ❖ To Motivate students developing projects using digital signal processing techniques by avoiding programming difficulties.
- ❖ To increase the integration of an algorithm's results in a graphic environment of simple programming.
- ❖ To Favour the creativity of students when carrying out the projects and dissertations necessary for their university degree.
- ❖ To allow communication between the most well-known programming languages (Java, C++ and C#) and Octave.
- ❖ To publish the project at Sourceforge.net, for its use and assessment by the scientific community.

3. METHODS

The main technologies used in the development of the proposed APIs are detailed below.

3.1 Octave

Octave [1][2] is a high-level language for numerical calculation, whose syntax is compatible with Matlab, but is developed by the free software community.

What makes Octave different from other programming languages?

Octave is particularly oriented towards the scientific world. Among its main differences from other programming languages, the following stand out:

1. Native matrix operation.
2. Native operation with complex numbers.
3. Language is interpreted.

These characteristics mean that scientific algorithms can be developed in a far shorter time than in other programming languages. Therefore, Octave is the ideal language for the development of digital signal processing algorithms, digital image processing, control systems, statistics, etc.

Furthermore, there a great many toolboxes that allow the user to avoid having to start from scratch when wishing to deal with a particular subject matter.

3.2 C++

C++ [3] is a programming language designed by Bjarne Stroustrup in the mid 1980s as an extension to the C programming language.

C++ is regarded by many as being the most powerful language, due to the fact that it allows the operator to work at both high and low levels. However, at the same time, it is one that bears the least number of automations (as with C, almost everything has be done manually), which makes it difficult to learn. The following are some of its main characteristics:

- Programming is object oriented
- Portability
- Brevity
- Modular programming
- Speed

3.3 C#

C# (C Sharp) [4] is the new general-use language designed by Microsoft for its .NET platform [5]. Its syntax and structure are very similar to that of C++. However, its straightforwardness and high degree of productivity are comparable to that of Visual Basic. The following are some this language's characteristics:

- Simplicity.
- Modernity.
- Object-oriented.
- Efficiency.

3.4 Java

Java is an object-oriented programming language developed by James Gosling and colleagues at Sun Microsystems in the early 1990s. The language, which was designed to be platform independent, is a derivative of C++ with a simpler syntax, a more robust runtime environment and simplified memory management.

Operating on multiple platforms in heterogeneous networks invalidates the traditional schemes of binary distribution, release, upgrade, patch, and so on. To survive in this jungle, the Java programming language must be architecture neutral, portable, and dynamically adaptable.

4. DESIGN

Throughout their degree studies, engineering students (more specifically those specializing in electronics and telecommunications) develop a great number of algorithms for digital signal processing in Octave/Matlab.

So as to avoid that students can reuse the codes of the algorithms developed in instead of codifying them again, and also to provide the resulting software with an attractive interface, the implementation of the "joPAS", "coPAS" and "monoPAS" APIs was proposed.

The communication among the bridge-languages in each case is just based in three main classes:

- **"ParserOctave" class**. It is the main API class; it manages the communication with Octave. It is responsible for loading the Octave thread and administering their input and output through the control of windows API functions. The streams are redirected to input/output stream descriptors which send the Octave instructions ("execute" function) and get the results in the appropriate variable ("executeAndSave" function).
- **"JoPAS", "Copas"or "Monopas" classes**. In each of these classes the Parser, defined previously, is started and closed. Also, the Octave's algorithm is programmed with the according execution form, saving the points of X and Y axis.
- **GUI class**. This class is based on a plot. With a button, "JoPAS", "Copas" or "Monopas" class is now launched. But, the real issue is to implement a correct library (depending of the language Java, C++ or C#), the graphic is represented, using the X and Y axis.

Plainly, the structure has been simplified in such a way that the knowledge of these three classes is enough for these APIs. This is one reason to reach the education objective.

As the scientific calculation is still carried out in Octave, the APIs allow the exchange of variables between the languages, as well as the execution of Octave commands from C++, C or Java#. The work methodology with "coPAS", "monoPAS" or "joPAS" would be as it is showed in Figure 1.

When this has been carried out, the Parser execute the Octave's algorithms. Usually this is a combination of several instructions just to execute, and others instructions which their results have to be saved. This is possible thanks to exchange the variables between the bridge-languages. Once the execution of the algorithm has been concluded, the results are stored and the variables are turned back into the language of C# (C++ or Java). Having reached this point, it will be the compiled language that undertakes the graphic representation of the result.

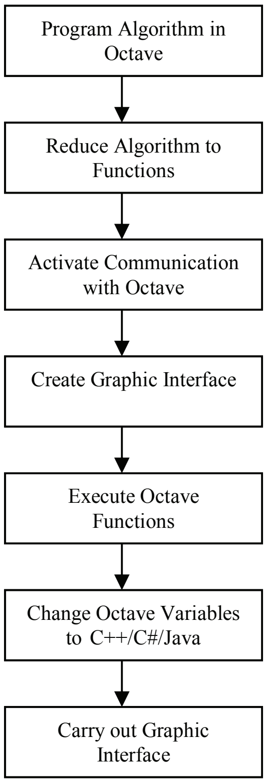
Program Algorithm in Octave
Reduce Algorithm to Functions
Activate Communication with Octave
Create Graphic Interface
Execute Octave Functions
Change Octave Variables to C++/C#/Java
Carry out Graphic Interface

Figure 1 . Design methodology flow chart

4. RESULTS

The implementation of digital signal processing applications using "joPAS", "coPAS" and "monoPAS" is extremely straightforward. Below, it is explained Butterworth Filter demo programmed with the presented APIS.

The first step is delimit the Octave instructions, or sentences, for carrying out the calculate (figure 2).

```
N=4;
W=0.5;
[B,A]=BUTTER(N,W);
[H,F]=freqz(B,A,512,20000);
modulodB=20*log10(abs(H));
```

Figure 2. "Octave" instructions for a filter.

This code is included in a C++, C# or Java application with in a function which obtains the number of points, the value of the points in the X axis and in the Y axis. Previously to this, the parser must be initiated. In an easy way, the code for this signal is described in the following figures 3, 4 and 5. Graphical result can be seen in figure 6.

```
//Function to Execute Octave's instructions
void Copas::octaveAlgorithm(double** ejeX,double**
ejeY,int* puntos){
//Executes the Octave commands using one static variable
to the ParserOctave class (p)
Copas::p->execute("N=4;\n");
Copas::p->execute("W=0.5\n");
Copas::p->execute("[B,A]=BUTTER(N,W);\n");
Copas::p->execute("[H,F]=freqz(B,A,512,20000);\n");
Copas::p->execute("modulodB=20*log10(abs(H));\n");
//Write the values at the output variables for the X axis
Copas::p->executeAndSave("ejeX=F'\n", "ejeX");
while (ParserOctave::m == NULL);
double* auxX = (ParserOctave::m->getReal())[0];
*ejeX = new double[ParserOctave::puntos];
for(int i = 0; i<ParserOctave::puntos; i++)
(*ejeX)[i] = auxX[i];
//Write the values at the output variables for the Y axis
Copas::p->executeAndSave("ejeY=modulodB'\n","ejeY");
while (ParserOctave::m == NULL) ;
double* auxY = (ParserOctave::m->getReal())[0];
*ejeY = new double[ParserOctave::puntos];
for(int i = 0; i<ParserOctave::puntos; i++)
(*ejeY)[i] = auxY[i];
//Write the values at the output variables for the point's
number
*puntos = ParserOctave::puntos;}
```

Figure 3. "Octave" instructions for a filter using "coPAS".

Not only the technical results have been taken into account, but also those obtained from the students' learning process. In the table below, the results from a satisfaction survey given to the students are shown. Generally speaking, the results from a significant sampling of 35 students have been outstanding.

Each item has been evaluated between 1 and 10 points:

9-10: Strongly agree
7-8: Agree
5-6: Neutral
3-4: Disagree
1-2: Strongly Disagree

In table 1, it can be seen that the item which obtained the worst mark was number 3. This is because of the fact that the students not only have to control this tool, Octave code too. Other interesting thing that it can be seen in table 1 is that the students prefer "joPAS", it is because the learn Java language previously. But the average result is very satisfactory in general terms.

QUESTION ASKED (To a group of 35 students)	**joPAS**	**coPAS**	**monoPAS**
Is the documentation on the gateway clear?	9	8	8
Do the gateways cover all the operations of signal processing?	9	8,5	8,5
Degree of difficulty or time needed to master gateways?	8,5	8	8,5
Degree of difficulty or time required to develop digital signal processing systems in [Java, C or .NET]?	9,5	7	8
Does the gateway design allow one to go deeply into the subject content?	8,5	8	8
Do you find the gateways more motivating/easier to use than the traditional method?	8	8	8
General satisfaction	8,5	8	8

Table 1. Satisfaction poll results.

```
//Function to Execute Octave's instructions
static public void octaveAlgorithm(ref double[] ejeX, ref
double[] ejeY) {
//Executes the Octave commands using one static
variable to the ParserOctave class (p)
p.execute("N=4;\n");
p.execute("W=0.5;\n");
p.execute("[B,A]=BUTTER(N,W);\n");
p.execute("[H,F]=freqz(B,A,512,20000);\n");
p.execute("modulodB=20*log10(abs(H));\n");
//Write the values at the output variables for the X axis
p.executeAndSave("ejeX=F\n","ejeX");
while (ParserOctave.m == null);
ejeX = (double[])(ParserOctave.m.getReal())[0].Clone();
//Write the values at the output variables for the Y axis
p.executeAndSave("ejeY= modulodB '\n","ejeY");
while (ParserOctave.m == null)
ejeY = (double[])(ParserOctave.m.getReal())[0].Clone();}
```

Figure 4. "Octave" instructions for a filter using "monoPAS".

6. CONCLUSIONS

Finally, it is appropriate to highlight the benefits obtained from using the APIs for the design of signal processing applications requiring a graphic user interface. These advantages are, on the one hand, that by means of these APIs the implementation of the digital processing signal algorithm can be carried out in "OCTAVE", a suitable processing language for this task. On the other hand, it enables the implementation of the user interface in JAVA, C++ or C#, which are the most appropriate programming language for this kind of task.

By dividing the process into the separate algorithm and visualisation parts, the applications can be designed more quickly, the APIs being the element enabling this link to be carried out.

```
//Executes the Octave commands using local variables
jopas.Execute("N=4;");
jopas.Execute("W=0.5");
jopas.Execute("[B,A]=BUTTER(N,W);");
jopas.Execute("[H,F]=freqz(B,A,512,20000);");
jopas.Execute("modulodB=20*log10(abs(H));");
/* XYPlots of "F" and "modulodB", the plot title is "Módulo", the
* X label is "Frecuencia (Hz)" and Y label is "Módulo (dB)".*/
this.jopasLabel2.paintLabel("F", "modulodB",
"Módulo","Frecuencia (Hz)", "Módulo (dB)");
```

Figure 5. "Octave" instructions for a filter using "joPAS".

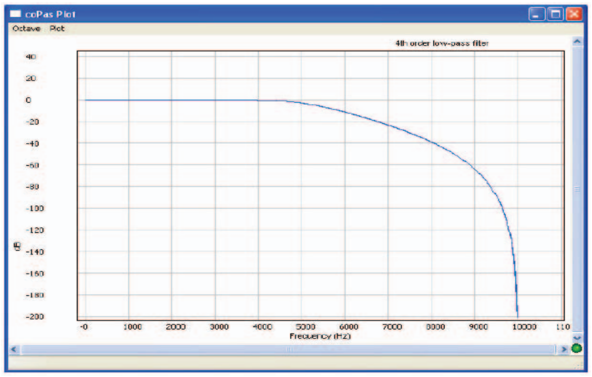

Figure 6. 4th order low-pass filter representation using "CoPAS"

7. ACKNOWLEDGEMENTS

The authors of this paper would like to thank the University of Deusto for the support it gives to this kind of initiative through the concession of a pedagogical innovation project in 2007.
This research was partially carried out under grant TEC2006-12887-C02-02 from the Ministry of Science and Technology of Spain.

8. REFERENCES

[1] Kurt Hornik, Friedrich Leisch, Achim Zeileis, "Ten Years of Octave Recent Developments and Plans for the Future" in Proc. DSC 2004, Vienna, Austria, 2004.
[2] B.L. Sturm, J.D. Gibson, "Signals and Systems using Matlab: an integrated suite of applications for exploring and teaching media signal processing", in Proceedings 35th Annual Conference Frontiers in Education, FIE '05.2005
[3] B. Stroustrup, "The C++ Programming Language" Addison Wesley, Special Edition, 2002.
[4] K. Watson. "Beginning C#". Wrox. 2001 Press.
[5] B. Powell and R. Weeks. "C# and the .NET framework: the C++ perspective" Indianapolis, Indiana: Sams, 2002.