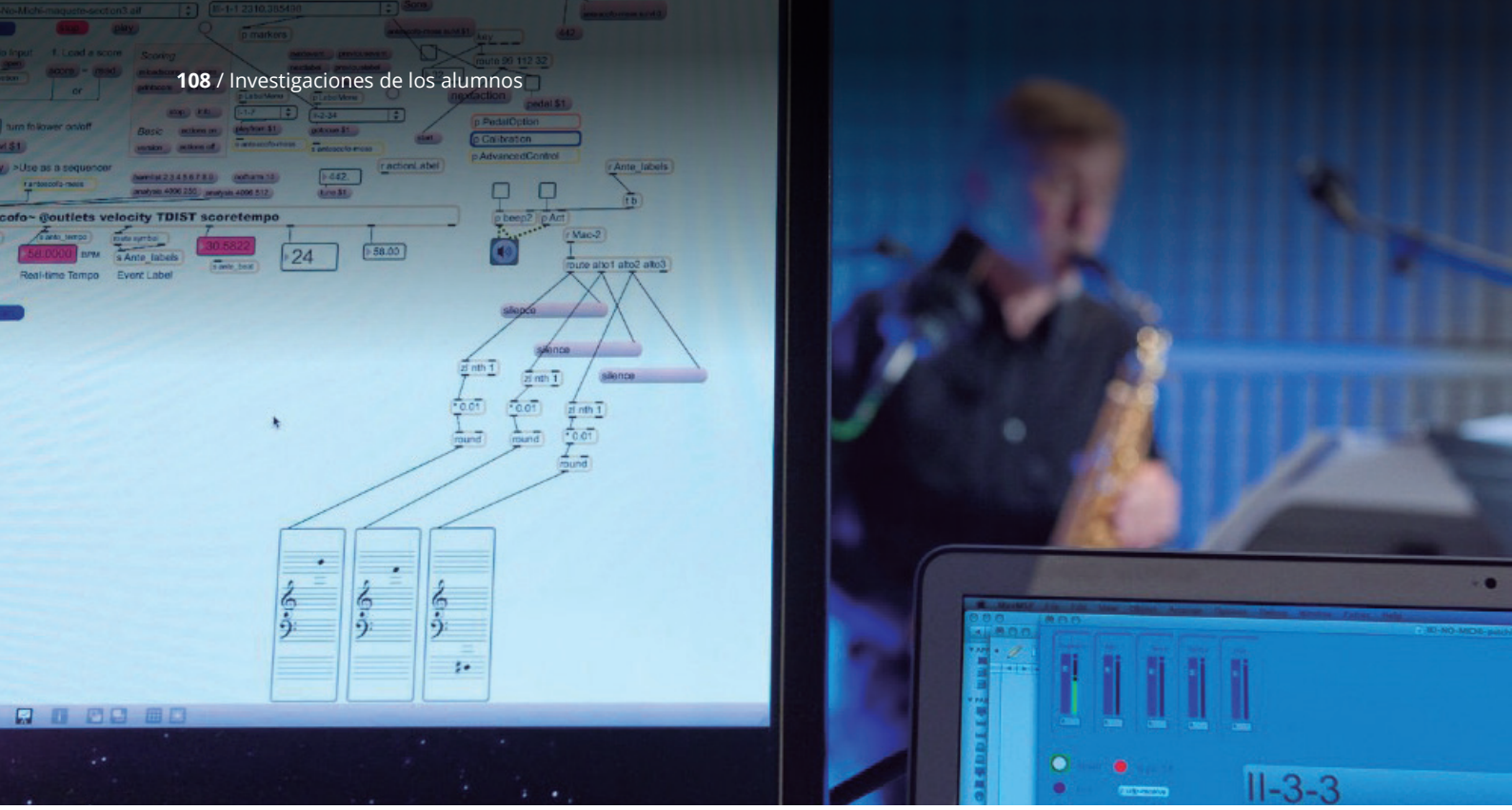

Primeros pasos en la compilación de procesadores modulares de audio en tiempo real en Antescofo

First steps toward embedding real-time audio computing in Antescofo

Nicolás Schmidt[1*], Alumno de 6to año
Arshia Cont[2], Investigador Guía
Jean-Louis Giavitto[3], Investigador Guía

[1]Departamento de Ciencia de la Computación, Escuela de Ingeniería, Pontificia Universidad Católica de Chile
[2]MuTanT Team-project, INRIA, Institut de Recherche et Coordination Acoustique/Musique
[3]MuTanT Team-project, INRIA, CNRS, Institut de Recherche et Coordination Acoustique/Musique
*Autor de correspondencia: nschmid1@uc.cl

Nicolás Schmidt[1*], 6th year student
Arshia Cont[2], Head of Research
Jean-Louis Giavitto[3], Head of Research

[1]Department of Computer Science, Engineering School, Pontificia Universidad Católica de Chile
[2]MuTanT Team-project, INRIA, Institut de Recherche et Coordination Acoustique/Musique
[3]MuTanT Team-project, INRIA, CNRS, Institut de Recherche et Coordination Acoustique/Musique
* Corresponding author: nschmid1@uc.cl

RESUMEN

El problema de la preservación en el tiempo de piezas de música interactiva ha estado siempre presente en este género de la música contemporánea. Debido al tamaño, versatilidad y grandes recursos computacionales del mini computador UDOO, el equipo MUTANT del centro de investigación INRIA decidió compilar una versión del software Antescofo para este hardware con el fin de poder almacenar y preservar estas piezas en el tiempo. Sin embargo, debido principalmente a la arquitectura de procesamiento de señales digitales del software, se pudo observar un desempeño no óptimo de la aplicación para interpretación en tiempo real de piezas interactivas. En este artículo se presenta una versión de Antescofo que cambia el paradigma de procesamiento de los efectos y sonidos, desde una arquitectura de sistema de paso de mensajes entre Antescofo a patches externos, hacia una arquitectura que integra la librería Faust de procesamiento de señales. La finalidad de este cambio es compilar una versión para la plataforma UDOO que permita ejecutar las piezas sin problemas de rendimiento. Esta investigación se basa en la realización de tests de performance para ambas versiones buscando esclarecer si la nueva arquitectura planteada por MUTANT posee efectivamente un mejor desempeño en aplicaciones en tiempo real que la versión anterior. La metodología utilizada consiste en la comparación de dos versiones adaptadas de la obra "Anthèmes 2" de Pierre Boulez y la comparación de los tiempos de ejecución del proceso encargado del procesamiento de señales digitales. Con la nueva arquitectura se obtuvo una mejora de un 46%. Los resultados expuestos permiten concluir que las características técnicas del mini computador UDOO hacen posible crear una versión de Antescofo para esta plataforma. Así, esta nueva plataforma permitirá a la comunidad de músicos contemporáneos preservar sus piezas musicales en el tiempo.

Palabras clave: Antescofo, UDOO, faust, preservación de música interactiva, música interactiva.

ABSTRACT

The problem of preservation through time of interactive music pieces has always been present in this field of contemporary music. Because of the size, versatility and computational power of the UDOO minicomputer, INRIA's MUTANT team has decided to compile a version of the Antescofo software in order to conserve these musical pieces. However, because of the digital signal processing architecture of the software, a non-optimal performance for the interpretation of real time pieces was observed. In this article, we present a new version of Antescofo that changes the signal processing paradigm from an architecture of a message passing system between Antescofo and external patches, to an architecture that integrates the Faust digital signal processing library. This research is based on the perform of profiling tests for both versions, with the aim of seeing if the new architecture is actually better in terms of performance for real time applications than the older version. The methodology used was to compare both versions performing the interactive music piece "Anthèmes 2" from Pierre Boulez and the comparison of the execution times of the digital signal processing process. The new architecture showed an improvement of 46%. The results exposed suggest that the technical resources of the UDOO platform are enough to run a version of Antescofo, allowing this way to preserve interactive musical pieces through time.

Keywords: Antescofo, UDOO, faust, interactive music preservation, interactive music.

1. INTRODUCCIÓN

Existe un campo en música contemporánea llamado música interactiva. La música interactiva corresponde a la composición de una pieza escrita con partes para uno o varios intérpretes humanos y para componentes electrónicos. En estas piezas, los componentes electrónicos responden a la ejecución del músico, generándose un *feedback* mutuo en el que ambas partes interpretan la melodía.

Dentro de las principales plataformas para la composición de música interactiva se destacan PureData y Max MSP. Ambas herramientas son *softwares* de programación gráfica basado en el ensamble de módulos o *patches*,

1. INTRODUCTION

A field called interactive music exists within the realm of contemporary music. Interactive music corresponds to the composition of a piece written with parts for one or various human performers and for electronic components. In these pieces, the electronic components respond to the performer's execution, generating a mutual feedback in which both parts interpret the melody.

Among the main platforms for the composition of interactive music, PureData and Max MSP stand out. Both tools are graphic programming software based on the ensemble of modules or patches, which consist of

que consisten en sub-programas que poseen funciones específicas. Las principales funcionalidades de estas plataformas están orientadas al procesamiento de sonido e imágenes en tiempo real; sin embargo, hasta el año 2007 no existía un *patch* especializado en resolver el problema de la sincronización entre humano y máquina en la ejecución de música interactiva.

Antescofo es un *software* creado por Arshia Cont y el compositor Marco Stroppa en el 2007. Su desarrollo ha sido continuado por el equipo **MUTANT** del centro de investigación **INRIA**, compuesto por personas del **IRCAM**, INRIA y del **CNRS** desde el 2012. El objetivo principal del *software* es lograr la sincronización entre los componentes electrónicos y el músico en la ejecución de la pieza. El *software* permite al compositor escribir tanto la partitura del músico como las acciones que deben ser gatilladas por la máquina frente a los eventos musicales.

Antescofo está compuesto por un sistema de **machine listening** y un sistema de ingeniería reactiva [1]. El primero es responsable del procesamiento del audio y reconocimiento de diversos eventos y parámetros musicales en tiempo real. El Sistema de ingeniería reactiva es responsable de gatillar las acciones especificadas en la partitura, como respuesta a ciertos eventos producidos por el músico, tal como se muestra en la Figura 1.

Antescofo ha sido compilado para el sistema operativo Mac OS X, tanto en la plataforma PureData como en Max MSP, mientras que en Linux solamente existe una versión para PureData. Actualmente el equipo MUTANT se encuentra trabajando en la compilación de una versión para PureData de Windows, lo cual permitirá que más músicos contemporáneos puedan acceder y utilizar esta herramienta para sus composiciones.

sub-programs that have specific functions. The main functionality of these platforms is focused on the processing of images and sounds in real time; nevertheless, until 2007, there was no patch specialized in solving the problem of synchronization between human and machine in the execution of interactive music.

Antescofo is a software created by Arshia Cont and the composer Marco Stroppa in 2007. Its development has been continued by the **MUTANT** team at the **INRIA** research center, by staff members of **IRCAM**, INRIA and **CNRS** since 2012. The main objective of the software is to achieve synchronization between the electronic components and the musician when executing the piece. The software allows the composer to write both, the musician's score as well as the actions that must be triggered by the machine before the musical events.

Antescofo consists of a **listening machine** system and a reactive engineering system [1]. The first is responsible for processing audio and recognizing different events and musical parameters in real time. The reactive engineering system is responsible for triggering the actions specified in the score, such as the response to certain events produced by the musician, as shown in Figure 1.

Antescofo has been compiled for Mac OS X operating system for both; PureData and Max MSP platforms, while in Linux it only exists for PureData platform. Currently, the MUTANT team is working on the compilation of a new version of Antescofo for PureData for Windows, which will allow more contemporary musicians to access and use these tools for their works.

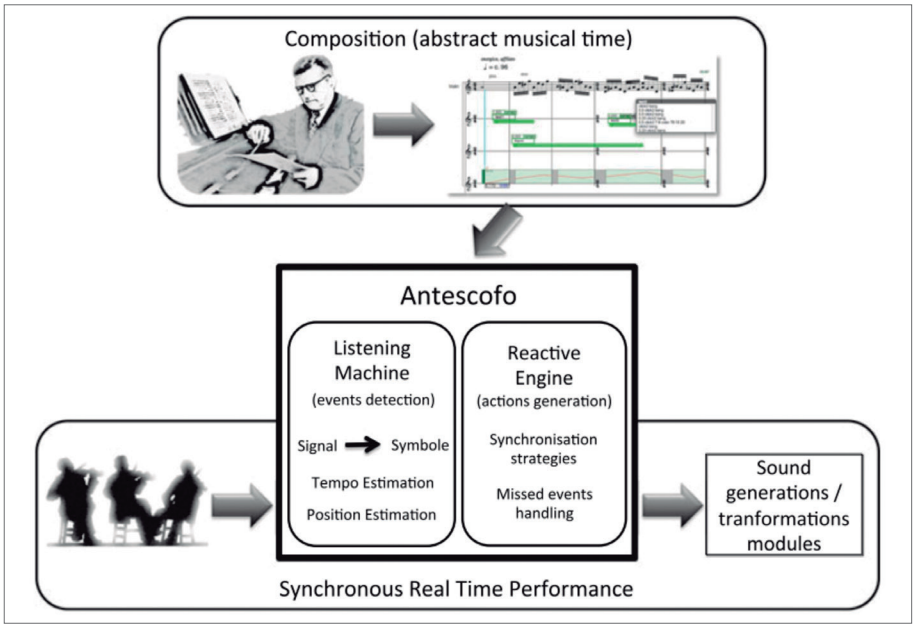

Figura 1. Proceso de funcionamiento de Antescofo en la recepción y generación de sonidos.

Figure 1. Antescofo's working process in the reception and generation of sound.

Independientemente de la cantidad de plataformas que permiten desarrollar este tipo de composiciones, la música interactiva está enfrentando un problema mayor: la preservación en el tiempo de las piezas musicales compuestas hoy. Como se ha visto en las últimas décadas, la tecnología, plataformas y protocolos de comunicación entre aplicaciones están cambiando continuamente. Esto, en un contexto artístico implica que las piezas musicales están susceptibles a quedar obsoletas al desaparecer la tecnología con la cual fueron desarrolladas, por lo que existe el riesgo de que el arte se pierda en el tiempo.

Frente a este desafío, se comienza un trabajo de compilación de una versión de Antescofo para PureData corriendo en un mini computador. Utilizar este tipo de *hardware* permitiría tener físicamente un mecanismo de almacenamiento y reproducción de piezas de música interactiva. Así, independiente de los cambios tecnológicos que puedan surgir en el futuro, se preserva la música y se evita la obsolescencia de la obra. Sin embargo, debido a las limitantes de *hardware* de los mini computadores en términos de recursos computacionales (poder de procesamiento, memoria y arquitectura), la versión de Antescofo compilada para esta plataforma no es capaz de procesar toda la información necesaria y tener una buena *performance* para una interpretación de una pieza de música interactiva en tiempo real.

Este trabajo busca realizar modificaciones en la arquitectura del *software* Antescofo con el objetivo de optimizar la cadena de generación de sonidos y procesamiento de efectos, procesos que son los principales causantes del mal desempeño en tiempo real. Los cambios en la arquitectura corresponden a relegar funciones de procesamiento de señales digitales a librerías especializadas compiladas en el mismo *software*. Luego de la implementación de esta arquitectura, se realizó un análisis comparativo entre ambas versiones de Antescofo con el fin de evaluar si la implementación de herramientas modulares para el procesamiento de señales digitales era efectivamente mejor en términos de rendimiento y de *performance* en tiempo real.

En este artículo se explica el procedimiento utilizado para acercarse a una nueva versión de Antescofo, la cual tiene por objetivo poder ser compilada y ejecutada en el mini computador UDOO de forma eficiente, y permitiendo además brindar nuevas funcionalidades al compositor. En un principio se detalla la arquitectura actual del procesamiento de señales digitales junto con los principales problemas asociados a ella. Luego, se presenta la nueva arquitectura propuesta, basada en la utilización de herramientas modulares de procesamiento de efectos y sonidos. A continuación, se detalla la comparación en pruebas de *performance* entre ambas versiones implementadas mediante la ejecución de la misma pieza musical, adaptada para ambas versiones. Finalmente, se explican los resultados y sus implicancias en la compilación y adaptación de una versión del *software* para UDOO, con el fin de preservar las obras de música interactiva a través del tiempo.

Independent from the amount of platforms that allow this kind of compositions to be developed, interactive music faces a bigger challenge: the preservation of musical pieces composed today throughout time. As it has been observed in the past few decades, technology, platforms and communication technology among applications are constantly changing. This, within an artistic context, implies that musical pieces are susceptible to become obsolete as the technology in which they were developed disappears, so there is a risk that art will be lost in time.

In light of this challenge, the task of compiling a version of Antescofo for PureData running on a mini computer begins. Using this type of hardware would make possible to have a physical mechanism of storage and reproduction of interactive musical pieces. Thus, despite the technological changes that may occur in the future, the music is preserved and the risk of it becoming obsolete is avoided. Nevertheless, due to the minicomputer hardware's limitations in terms of computational power (processing power, memory and architecture), the version of Antescofo compiled for this platform is not capable of processing all the necessary information, although has a good performance when interpreting a piece of interactive music in real time.

This work seeks to make modifications to the Antescofo's software architecture with the goal of optimizing the sound generation and effect processing chain, processes that are the main causes of poor performance in real time. The changes in architecture consist of relegating digital signal processing functions to specialized libraries compiled in the same software. After implementing this architecture, a comparative analysis between both Antescofo versions was performed to determine if the implementation of modular tools for the processing of digital signal was effectively better in terms of efficiency and performance in real time.

This article explains the procedure used to approach a new version of Antescofo, whose objective is to be compiled and executed in the UDOO minicomputer in an efficient manner and also to bring new functionalities to the composer. In the beginning, the actual architecture of the digital signal processing, along with the problems associated with it, is described. Then, the new proposed architecture, based on the utilization of modular sound and effects processing tools, is presented. Next, the comparison in performance tests between both versions implemented through the execution of the same musical piece adapted for both versions is depicted. Finally, the results and their implications in the compilation and adaptation of a software version for UDOO, with the goal of preserving works of interactive music throughout time, are explained.

2. METODOLOGÍA

La actual cadena de generación de sonidos y procesamiento de efectos de Antescofo comienza con la carga de la partitura, luego el sistema de *machine listening* escucha al músico, esperando la aparición de las acciones especificadas en la partitura. Frente a la aparición de un evento que corresponde al componente electrónico, se envía un mensaje a la plataforma (PureData o Max MSP), y a partir de los parámetros del mensaje, se genera o procesa el sonido en *patches* externos a Antescofo. Esto puede verse claramente en la Figura 2.

El actual paradigma de procesamiento de sonidos tiene los siguientes problemas:

- Desempeño no óptimo: Como la generación de sonidos y procesamiento de efectos depende de la plataforma en la que está corriendo Antescofo (PureData o Max MSP), la eficiencia del software depende mucho de cómo la plataforma maneja los recursos computacionales. Max MSP y PureData no manejan los recursos de forma óptima para el procesamiento de sonidos en tiempo real, y la arquitectura actual impide tener un flujo de información y optimización de la ejecución del *software*. El sistema de paso de mensajes no permite una optimización preventiva del desempeño del software, ya que sólo se sabe qué es lo que se va a procesar una vez que se gatilla el mensaje desde Antescofo [2].
- Dependencias de la plataforma: Como el procesamiento de sonidos y efectos es procesado por la plataforma (Max MSP o PureData), se tienen muchas dependencias de la misma. Esto es un problema debido a que si la plataforma cambia de una versión a la siguiente, la pieza musical compuesta para Antescofo podría dejar de

2. METHODOLOGY

Antescofo's current sound generation and effect processing chain begins with the loading of the score; then, the machine listening system listens to the musician, waiting for the actions specified in the score to appear. When a new event that corresponds to the electronic component appears, a message is sent to the platform (Pure Data o Max MSP), and depending on the message parameters, a sound is processed or generated in external patches to Antescofo. This can be clearly observed in Figure 2.

The current sound processing paradigm has the following problems:

- Non-optimal performance: Since sound generation and effect processing depend on the platform on which Antescofo is executed (PureData or Max MSP), the efficiency of the software depends considerably on how the platform manages the computational power. Max MSP and PureData do not manage computational power in a form that is optimal for the processing of sound in real time, and the current architecture hinders the flow of information and the optimization of the software's execution. The message passing system does not permit a preventive software performance optimization, as what will be processed is only known once the message is triggered from Antescofo [2].
- Platform dependencies: Since sound and effect processing is processed by the platform (Max MSP or PureData), there are many dependencies from it. This is a problem because, if the platform changes from a version to the next, the musical piece composed for Antescofo could stop working and lose much of its electronic components. This has great implications in the problem of preservation of interactive musical pieces in time.

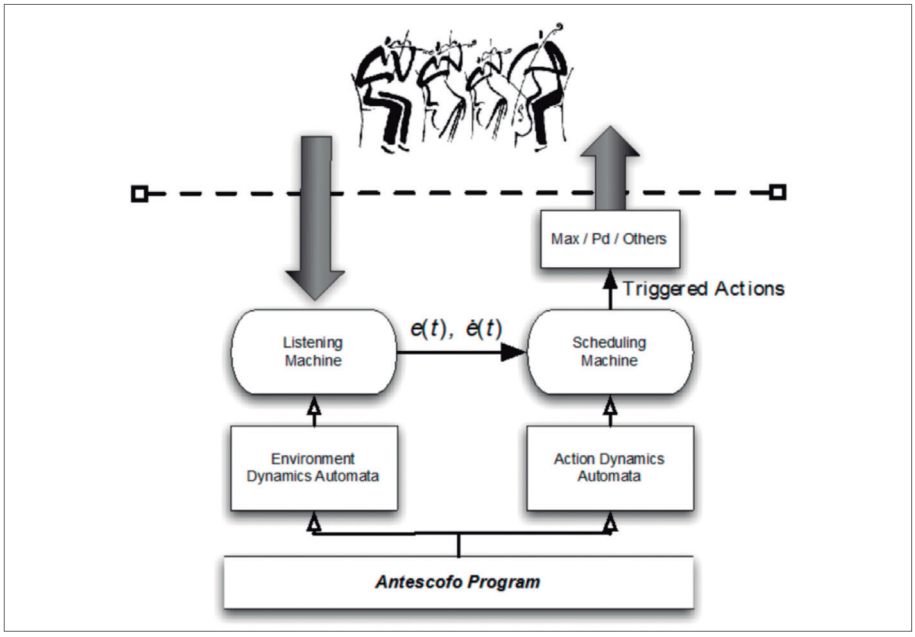

Figura 2. Metodología se sincronización de Antescofo en la gatillación de las acciones y paso de mensajes a la plataforma.

Figure 2. Antescofo's synchronization methodology in the triggering of action and message passing to the platform.

funcionar y perder gran parte de sus componentes electrónicos. Esto tiene gran implicancia en el problema de la preservación de las piezas de música interactiva en el tiempo.

Como solución a estos problemas, se plantea una nueva arquitectura del software para la generación de sonidos y procesamiento de efectos. Esta nueva forma de procesar los sonidos cambia el paradigma desde el sistema de paso de mensajes desde Antescofo hacia los otros *patches* de la plataforma, a una metodología en la que Antescofo pasa a ser el principal responsable de esta función, mediante la integración de herramientas modulares de procesamiento de sonidos y efectos. De esta forma se obtiene un mejor control sobre los procesos de compilación de las funciones, no se depende tanto de la plataforma en la cual está corriendo el programa, y por sobre todo, se logra una mayor eficiencia y desempeño de la aplicación.

La integración de las herramientas modulares permite al compositor crear sus propios efectos dentro de la misma partitura. Además, el uso de estas herramientas hace que la partitura sea completamente auto-contenida, ya que deja de depender de la plataforma sobre la cual está corriendo. Esto significa que los efectos y forma en que los sonidos son procesados se mantendrán invariante independiente de los cambios que puedan experimentar las plataformas en un futuro.

Una de las herramientas modulares integradas en Antescofo es Faust, una librería desarrollada por **Grame** en el centro de investigación de INRIA ubicado en Lyon, Francia. Faust es un lenguaje de programación funcional para el procesamiento de señales en tiempo real. Este lenguaje permite la programación de efectos y procesadores de señales y compilarlos como aplicaciones *standalone*. Faust traduce desde un lenguaje de alto nivel descriptivo de señales de audio a un programa en C++ optimizado. El hecho de que se produzcan programas en C++ permite al usuario compilar el software para correr como plugins de Max, PureData, VST plugins y aplicaciones Alsa o Coreaudio [3].

La integración de Faust en Antescofo optimiza el proceso de procesamiento de señales digitales en tiempo real debido a que los efectos son pre-compilados a través de un compilador On-the-fly integrado en la librería. Debido a la integración de esta herramienta modular, la nueva arquitectura de la generación de sonidos queda entonces de la siguiente forma: al cargar la partitura en Antescofo se leen todos los efectos que se espera que se generen y se pre-compilan con el compilador On-the-fly que posee las librería. Estos efectos pre-compilados son pasados a Antescofo, el cual mediante su sistema reactivo simplemente llama a los efectos pasándole los parámetros respectivos al momento en que son gatillados por el músico, como se muestra en la Figura 3. De esta forma se evita el proceso de paso de mensajes y de compilación de los efectos en tiempo real, ahorrando tiempo que es fundamental para la ejecución de este tipo de piezas en tiempo real.

As a solution to this problem, we propose a new software architecture for generation of sound and effect processing. This new way of processing sounds changes the paradigm from the message passing system from Antescofo to other platform patches, to a methodology in which Antescofo becomes the main system responsible for this function through the integration of modular tools of sound and effect processing. In this manner, better control of the process of the compilation of the functions is obtained, as it does not depend so much of the platform on which the program is executed, and above everything else, better application efficiency and performance is achieved.

The integration of the modular tools allows the composer to create his or her own effects within the same score. In addition, the use of these tools makes the score completely self-contained, as it becomes independent of the platform on which it's executed. This means that the effects and the form in which the sounds are processed remain unchanged, despite any modifications that the platforms may experience in the future.

One of the modular tools integrated in Antescofo is Faust, a library developed by **Grame** at INRIA research center located in Lyon, France. Faust is a functional programming language for processing signals in real time. This language makes effects programming and signal processing possible, and to compile them as standalone applications. Faust translates audio signals from a high level descriptive language to a program optimized in C++. The fact that programs are produced in C++ allows the user to compile the software to run as plugins of Max, PureData, VST plugins, and Alsa or Coreaudio applications [3].

Faust integration in Antescofo optimizes the process of digital signal processing in real time, because the effects are pre-compiled through an on-the-fly compilator integrated to the library. Due to the integration of this modular tool, the new sound generating architecture works as follows: once the score is loaded in Antescofo, all the effects to be generated are read and pre-compiled by the library's on-the-fly compilator. These pre-compiled effects are passed to Antescofo, which through its reactive system, simply calls all the effects by passing the respective parameters at the moment triggered by the musician, as shown in Figure 3. This way the process of message passing and the effect compilation in real time is avoided, saving time that is essential for the execution of this type of pieces in real time.

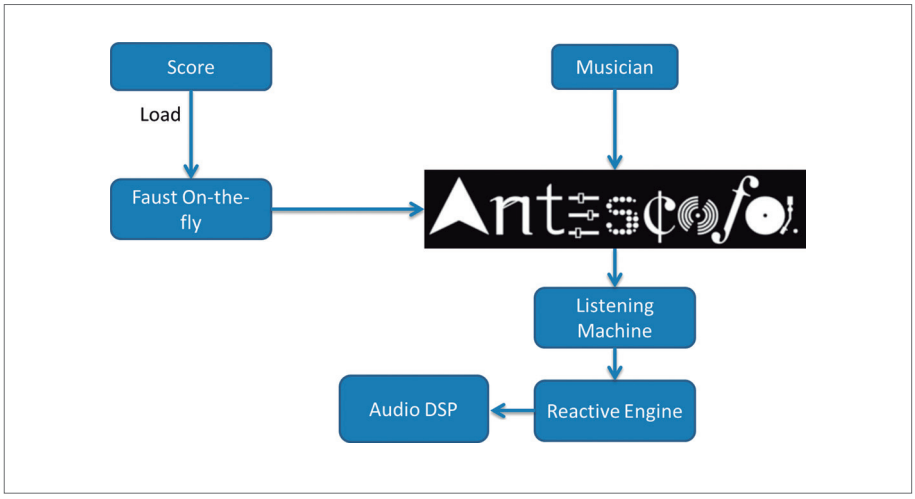

Figura 3. Cadena de generación de sonidos para la nueva versión de Antescofo utilizando la herramienta modular Faust.

Figure 3. Sounds generation chain for the new version of Antescofo using Faust modular tool.

Como la nueva versión de Antescofo creada por MUTANT integra una nueva arquitectura para la generación de sonidos y efectos, se espera que el desempeño de esta nueva versión en términos del tiempo computacional requerido para procesar la información mejore el desempeño del *software*. De esta forma se podría tener un mejor desempeño en sistemas de *hardware* con menores recursos computacionales, tal como lo es el mini computador UDOO.

Con el objetivo de poder hacer una correcta comparación de desempeño entre las dos versiones del *software* y poder aceptar o rechazar la hipótesis de que el cambio de arquitectura en el procesamiento de sonidos es mejor en la nueva versión que en la anterior, se procedió a escribir una versión de la obra "**Anthèmes 2"** de Pierre Boulez para la nueva versión de Antescofo. En esta se integraron en lenguaje Faust los componentes electrónicos que antes eran patches externos a Antescofo y coordinados mediante la arquitectura de mensajes. En la Figura 4 puede apreciarse el módulo de componentes electrónicos de la versión de Anthèmes 2 para la versión original de Antescofo. En esta puede verse que los efectos no pertenecen al *patch* de Antescofo, sino que están en *patches* externos, y que se comunican mediante el paso de mensajes.

Since the new version of Antescofo created by MUTANT integrates new sound and effect generation architecture, it's expected that the performance of this new version in terms of the computational time required to process the information improves the software's performance. In this way, better hardware system performance with less computational power could be attained, as with the UDOO minicomputer.

With the goal of making an accurate performance comparison between the two software versions, and to be able to accept or reject the hypothesis that the change in sound processing architecture is better in the newer version than the older one, a version of the Pierre Boulez's piece "**Anthèmes 2"** was written for the new version of Antescofo. The electronic components that had previously been external patches to Antescofo and had been coordinated through message architecture were integrated in Faust language. The electronic components module of the Anthèmes 2 version for the original version of Antescofo can be observed in Figure 4. This figure shows that the effects do not belong to Antescofo's patch, as they are in external patches and are communicated through message passing.

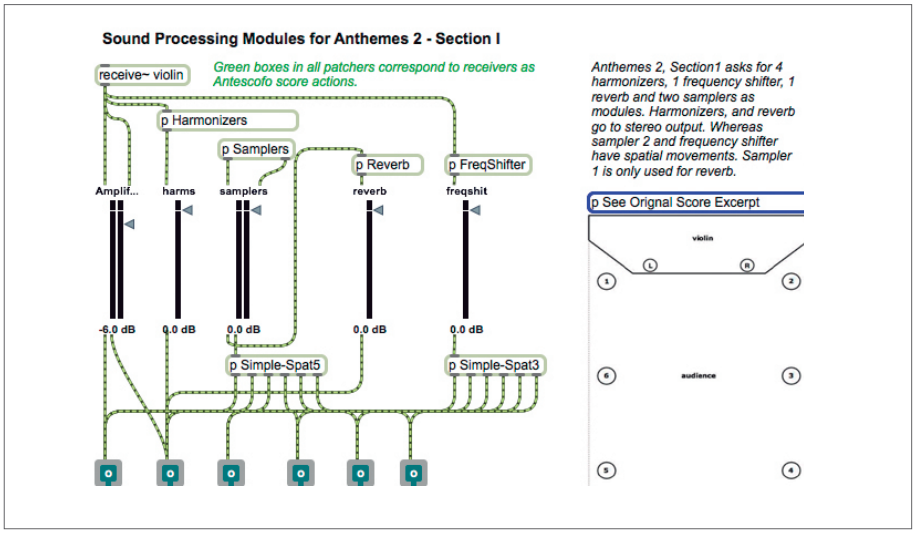

Figura 4. Patch de efectos de la pieza original "Anthèmes 2" adaptada por el IRCAM en Max MSP. Se pueden ver los componentes utilizados y la forma en la que fueron replicados.

Figure 4. Patch effects of original piece "Anthèmes 2" adapted by IRCAM in Max MSP. The components used and the form in which they were replicated can be seen here.

La implementación de los efectos en Faust utilizados en "Anthèmes 2" es la siguiente:

- Armonizador: implementado a partir de una modificación del código fuente de un *pitch shifter* implementado por Grame en lenguaje Faust. Puede ser encontrado en el **repositorio Git** de Faust.
- *Frequency shifter*: al igual que el armonizador, fue creado a partir de la modificación del código del *pitch shifter* de Grame.
- *Reverb*: este efecto fue creado a partir del efecto *freeverb*, también creado por Grame y disponible en el repositorio Git de Faust.
- *Panner*: Los dos *panners* son iguales, estos fueron implementados desde cero en lenguaje Faust y funcionan recibiendo el valor de la amplitud para cada uno de los 6 canales. Luego, utilizando elementos de discretización de parámetros de Antescofo se pudieron crear las curvas de panneo para cada uno de los canales.
- *Mixer*: se implementó además un mezclador en lenguaje Faust para poder enviar cada señal a su canal respectivo.

El *sampler* no pudo ser implementado con Faust debido a que esta herramienta no trabaja en el dominio espectral, solo temporal. De esta forma, para poder comparar ambas versiones de forma equitativa, se removió también el *sampler* de la versión original [4].

Luego de la implementación de los efectos Faust, la partitura de "Anthèmes 2" fue adaptada para responder a estas nuevas herramientas. El *patch* de "Anthèmes 2" original fue modificado para adaptar la nueva arquitectura que integra la herramienta modular Faust, de esta forma se obtuvieron las dos versiones para poder comparar el desempeño.

Para comparar el desempeño entre ambas versiones se utilizó la herramienta *Time Profiler* de las herramientas de desarrollo provistas por Xcode. *Time Profiler* permite al usuario grabar una secuencia de uso de un *software* y permite ver en detalle cuanto tiempo demora en ejecutar cada una de las funciones del código fuente en tiempo real. Se decidió utilizar esta herramienta porque permite ver en detalle cuánto tiempo y recursos computacionales fueron necesarios para poder lograr todos los procesos relacionados al procesamiento de efectos y sonidos. Las diferentes llamadas a los procesos son representadas en un diagrama de árbol, y todos los detalles, incluido el porcentaje de CPU utilizado por el proceso pueden ser vistos. Para este caso particular, la función de interés es la llamada "dspchain_tick", la cual es la encargada de todo el procesamiento de sonidos y efectos. Si se logra bajar el tiempo computacional de esta función con la nueva arquitectura, se tendrá un claro indicador de mejor desempeño.

Como Antescofo es una aplicación de audio en tiempo real, es muy difícil medir y comparar *performance* entre dos versiones. Esto se debe a que el principal objetivo de una

The implementation of the effects from Faust used in "Anthèmes 2" is as follows:

- Harmonizer: implemented from a modification of the source code of a pitch shifter implemented by Grame in Faust language. It can be found in Faust's **Git repository**.
- Frequency shifter: as the harmonizer, it was created from the modification of Grame's pitch shifter code.
- Reverb: this effect was created from the effect freeverb, also created by Grame and available in Faust's Git repository.
- Panner: Both panners are the same; these were implemented from scratch in Faust language and functions by receiving the amplitude value for each of the 6 channels. Then, it was possible to create panning curves for each one of the channels by using Antescofo's parameter discretization elements.
- Mixer: A mixer was also implemented in Faust language to send each signal to its corresponding channel.

The sampler could not be implemented with Faust because this tool does not work in the spectral domain, but only in the temporal. In order to be able to compare both versions with equity, the sampler was removed from the original version [4].

After the effects were implemented in Faust, the score of "Anthèmes 2" was adapted to respond to these new tools. The original patch of "Anthèmes 2" was modified to adapt the new architecture, which integrates Faust modular tool. Both versions were obtained this way to be able to compare performance.

The tool Time Profiler, from the developer tools provided by Xcode, was used to compare the performance of both versions. Time Profiler allows the user to record a sequence of software usage and to see in detail how much time it takes to execute each function of the source code in real time. This tool was chosen because it shows in detail how much time and how much computational power is needed to execute all the processes related to sound and effects. The different process calls are represented in a tree illustration, and all the details, including CPU percentage used by the process, can be seen therein. For this particular case, the function of interest is called "dspchain_tick", which is responsible for all sound and effect processing. A clear indication of better performance would be if the new architecture succeeds in reducing the computational time of this function.

Since Antescofo is a real time audio application, it is very difficult to measure and compare performance between both versions. This is because the main objective of a real time audio application is not to run as fast as possible, but to finish the computational processes before the arrival of the next audio buffer. To obtain a more accurate measure, a Max MSP mode that allows it to run as fast as possible by reading the entry audio sign from an audio file and not

aplicación de audio en tiempo real no es correr lo más rápido posible sino que terminar los procesos computacionales antes de la llegada del siguiente *buffer* de audio. Para lograr medir de forma más precisa, se seleccionó un modo de Max MSP que permite correr lo más rápido posible leyendo la señal de audio de entrada desde un archivo de audio y no desde una fuente de audio en tiempo real. De esta forma, la aplicación pasa a correr en un modo de *performance*, en cual lo que importa no es el desempeño en tiempo real sino que correr lo más rápido posible. En este modo es posible visualizar de mejor forma el tiempo computacional real tomado en cada proceso del *software*.

3. RESULTADOS Y DISCUSIÓN

Los resultados mostraron que el tiempo requerido para completar la ejecución del proceso "dspchain_tick" en el caso de la versión de Antescofo que maneja el procesamiento de sonidos mediante *patches* externos comunicados en base el paso de mensajes fue de 5900 ms. Por otra parte, el caso de la versión de Antescofo con cambio en su arquitectura utilizó 3150 ms. Este número representa la suma del tiempo que tomó en completarse la función "dspchain_tick" para todas las veces en que fue llamada durante el desempeño de la aplicación para una sección de "Anthèmes 2" de aproximadamente 1 min de duración.

Este número representa una mejora de un 46% con respecto a la versión original y es un claro indicador de que el cambio en la arquitectura en el caso de aplicaciones de audio en tiempo real mejora el desempeño global de la aplicación. En las Figura 5 puede verse la comparación en tiempo de ejecución entre ambas versiones de Antescofo, junto con los tiempos computacionales requeridos para concretar cada uno de los procesos. El proceso "dspchain_tick" aparece destacado.

from an audio source in real time was selected. In this way, the application goes on to run in performance mode, in which what matters is that it runs as fast as possible, not its performance in real time. By doing this, it is possible to get a better visualization of the real computational time of each of the software's process.

3. RESULTS AND DISCUSSION

The results show that the time required to complete the execution of the "dspchain_tick" process in the case of the Antescofo version that manages the sound process by means of external patches transmitted through message passing was of 5900 ms. On the other hand, the Antescofo version with a modification in its architecture used 3150 ms. This number represents the sum of the time that took to complete the function "dspchain_tick" for all the times it was called throughout the performance of the application during a section of "Anthèmes 2" lasting approximately 1 minute.

This number represents an improvement of 46% with respect to the original version and it is a clear indicator that the change in architecture in the case of real time audio applications improves the application's overall performance. Figure 5 shows the comparison in execution time between both Antescofo versions, along with computational times required to complete each process. The "dspchain_tick" process appears highlighted.

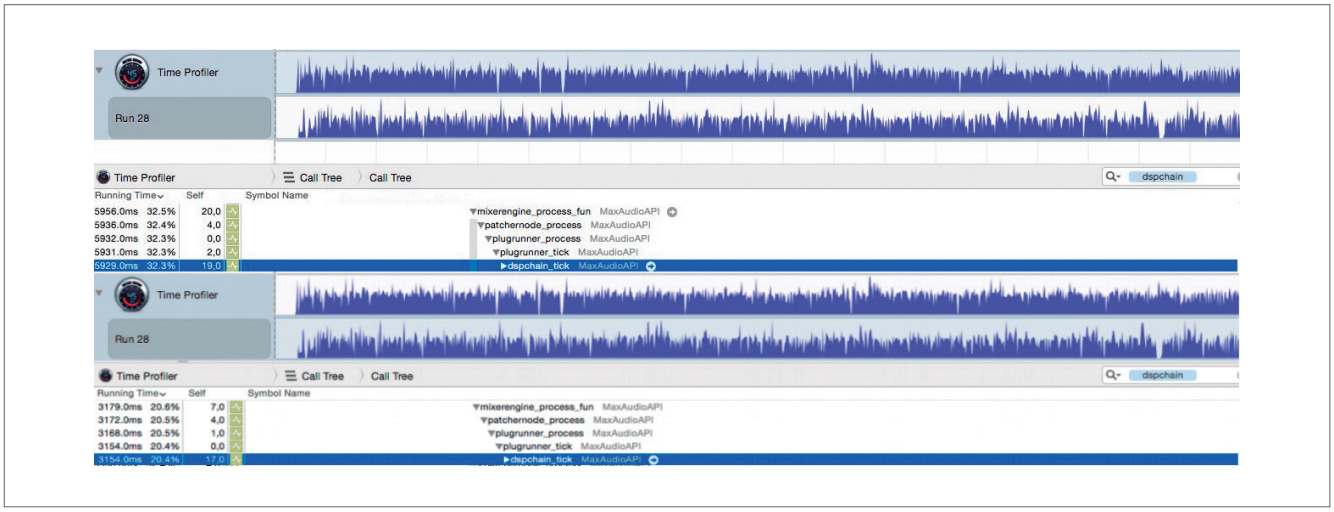

Figura 5. Time Profiler para la versión de Antescofo sin herramientas modulares (superior) y con ellas (inferior). Se puede apreciar la diferencia en los rendimientos de la misma pieza corriendo en ambas versiones de Antescofo.

Figure 5. Time Profiler for Antescofo version without modular tools (superior) and with them (inferior). The difference in performance while the same piece is running in both versions of Antescofo can be appreciated.

Como las pruebas fueron ejecutadas en un computador con sistema operativo Mac OS X, la parte de dependencia de la arquitectura de *hardware* no pudo ser medida en este *test*. Es por esta razón que las pruebas de *performance* en la plataforma UDOO aún están pendientes para cuando se tenga una versión más estable. Sin embargo, el resultado arrojado por el *Time Profiler* permite extender en análisis y aplicar los resultados de forma global. De esta forma, la mejora en *performance* de un 46% con respecto a la arquitectura anterior representa una mejora real en el desempeño del *software*, que puede ser extendida a otros sistemas y arquitecturas de *hardware* tales como el mini computador UDOO. Además permite asegurar que la nueva arquitectura de procesamiento de efectos y sonidos de forma modular es mejor que la que relegaba la responsabilidad a *patches* externos y se comunicaba mediante mensajes.

Con estos resultados, MUTANT comenzó el desarrollo de una nueva versión del algoritmo de *scheduling* de Antescofo, optimizado exclusivamente para la plataforma UDOO [5]. Los resultados mostrados en este artículo, junto al nuevo algoritmo de *scheduling* para UDOO, indican que eventualmente se podría correr una versión de Antescofo *standalone*. De esta forma, se podría tener una pieza de *hardware* preservada para el futuro, con las partituras de música interactiva pre-cargadas sin miedo a la posibilidad de perder información por obsolescencia de tecnología. Con la integración de las herramientas de procesamiento de sonidos modulares y el nuevo algoritmo de *scheduling*, el potencial del *software* crece y alcanza una nueva plataforma, que además permite, por su tamaño y potencial, preservar las piezas de música interactiva a través del tiempo.

4. CONCLUSIONES

A lo largo de este artículo se presentó uno de los principales problemas que presenta la música interactiva: la preservación de las piezas musicales en el tiempo. Se mostró el trabajo de MUTANT en la compilación de una versión del *software* Antescofo para el mini computador UDOO y se presentaron las conclusiones de esta nueva versión. Aquellas conclusiones revelaron que el principal problema en la ejecución de piezas en tiempo real utilizando Antescofo en UDOO se deben a la arquitectura de procesamiento de señales digitales.

En respuesta a lo anterior, se desarrolló la nueva arquitectura planteada por el equipo, la cual consideraba la integración de herramientas modulares de procesamiento de señales, específicamente la herramienta Faust. Se presentaron los beneficios de la utilización de esta nueva arquitectura en contraste con la anterior y se realizaron pruebas de desempeño a ambas versiones para poder tener un análisis comparativo. Con estos resultados se buscaba aceptar o rechazar la hipótesis de que la nueva arquitectura mejoraría

Since the tests were executed in a computer with Mac OS X operating system, the hardware architecture dependency part could not be measured in this test. This is the reason why the performance tests in UDOO platform are still pending until a more stable version is available. Nevertheless, the results obtained with Time Profiler allow analysis to expand and the results to be applied in an overall manner. In this manner, a 46% performance improvement with respect to the older architecture represents a real improvement in the performance of the software, which can be expanded to other systems and architectures such as the UDOO minicomputer. It also confirms that the new architecture of sound and effect processing in modular form is better than that which relegated the function to external patches and communicated through message passing.

With these results, MUTANT began developing a new version of Antescofo's scheduling algorithm, optimizing it exclusively for UDOO's platform [5]. The results shown in this article, along with the new algorithm scheduling for UDOO, indicate that a standalone version of Antescofo could be run eventually. In this manner, a piece of hardware could be preserved for the future, with interactive music scores pre-loaded without fearing the possibility of losing information due to obsolete technology. With the integration of modular sound processing tools and the new scheduling algorithm, the software's potential grows and reaches a new platform, which also allows, due to its size and potential, to preserve the pieces of interactive music throughout time.

4. CONCLUSIONS

Throughout this article, one of the main problems affecting interactive music was presented: the preservation of musical pieces in time. The work of MUTANT in compiling a version of Antescofo software for the UDOO minicomputer was shown and the conclusions of this new version were given. Those conclusions revealed that the main execution problem with the pieces in real time when using Antescofo in UDOO comes from the architecture of digital signal processing.

In response to this problem, new architecture proposed by the team was developed, which considered the integration of modular signal processing tools, specifically Faust. The benefits of using this new architecture as opposed to the older one were presented and performance test were conducted on both versions to obtain a comparative analysis. With these results, the objective was to accept or reject the hypothesis that the new architecture would improve the software's sound generation and effect processing performance.

el desempeño del *software* en la generación de sonidos y procesamiento de efectos.

Para llevar a cabo estas pruebas, se adaptó la pieza musical de Pierre Boulez "Anthèmes 2" para ambas versiones de Antescofo. Los efectos y procesadores de sonido fueron adaptados al lenguaje Faust y se reemplazó el sistema de paso de mensajes anterior. Se adaptaron los eventos de la partitura para responder a la nueva arquitectura y se llevaron a cabo *tests* de *performance*. Estas pruebas fueron realizadas a través del *software* XCode, con la herramienta *Time Profiler*, corriendo en la plataforma Max MSP en un sistema operativo Mac OS X.

Los resultados de estos *test* permiten concluir que el uso de herramientas de procesamiento de sonidos modulares embebidos en Antescofo optimiza el desempeño de la aplicación en términos del tiempo computacional requerido para procesar la información referente al procesamiento de señales en un 46%. Este resultado puede ser usado como parte de la optimización de la nueva versión de Antescofo para UDOO, en la que MUTANT está trabajando actualmente. Esta mejora permitirá que Antescofo pueda correr de forma *standalone* en el futuro, creando así una nueva plataforma de composición de música interactiva.

Esta investigación y sus resultados permiten concluir que las características técnicas del mini computador UDOO podrían hacer posible la creación de una versión de Antescofo para esta plataforma. Sin embargo, hasta no realizar las pruebas en la nueva versión del *software* corriendo en la plataforma mencionada, no se sabrá con certeza que una mejora en el rendimiento de la cadena de generación de sonidos de un 46% sea suficiente para que el *software* corra sin problemas. Sin embargo constituyen los primeros pasos para sentar las bases de la compilación de procesadores modulares de audio en tiempo real en Antescofo para la plataforma UDOO.

La creación de una versión de Antescofo para UDOO tiene como consecuencia importante el hecho de que esta nueva plataforma permitirá a la comunidad de músicos contemporáneos preservar sus piezas musicales a través del tiempo. De esta forma, las partituras de música interactiva creadas hoy podrán ser preservadas en este pequeño *hardware* y luego almacenarse para usos de preservación del arte. Esta se mantendrá intacta y permitirá su ejecución independiente de los cambios tecnológicos que puedan surgir.

Las pruebas de desempeño de ambas versiones de Antescofo en ambiente Max MSP y sistema operativo Mac OS X, revelaron un primer acercamiento a la optimización del desempeño de este *software* para la plataforma UDOO. Sin embargo, cuando se termine el nuevo algoritmo de *scheduling* para UDOO, se deberán volver a realizar pruebas de desempeño, de forma de tener resultados más precisos acerca de la optimización de Antescofo utilizando la nueva arquitectura de procesamiento de señales.

To conduct these tests, Pierre Boulez's musical piece "Anthèmes 2" was adapted for both versions of Antescofo. The effects and sound processors were adapted to Faust language and the older message passing system was replaced. The score's events were adapted to respond to the new architecture and performance tests were conducted. These test were performed through Xcode software and Time Profiler tool, running on Max MSP platform and Mac Os X operating system.

These test results lead to a conclusion that using modular sound processing tools embedded in Antescofo optimizes the performance of the application in terms of the computational time required to process information regarding signal processing by 46%. This result can be used as part of the optimization of the new version of Antescofo for UDOO, in which MUTANT is currently working. This improvement will allow Antescofo to run in standalone form in the future, thus creating a new platform for interactive music composition.

This research and its results lead to the conclusion that the technical characteristics of the UDOO minicomputer could make possible the creation of an Antescofo version for this platform. However, until further testing is done on the new version of the software running on the referenced platform, the question of whether a 46% improvement in the performance of the sound generation chain is sufficient for the software to run without problems *will* remain uncertain. Nevertheless, it constitutes the first steps to set the foundations of the compilation of Antescofo's modular sound processing in real time for UDOO's platform.

The creation of a new version of Antescofo for UDOO can offer a new platform that will allow the community of contemporary musicians to preserve their works throughout time. In this manner, the scores of interactive music created today could be preserved in this small hardware and later stored for art preservation purposes. The pieces will remain intact and their execution will be possible despite of any technological changes that may occur.

The performance tests of both Antescofo versions in Max MSP for Mac Os X operating systems first revealed an approach for optimizing this software for UDOO's platform. However, once the new scheduling algorithm for UDOO is ready, new performance testing will be conducted in order to obtain more precise results regarding Antescofo's optimization by using new signal processing architecture.

AGRADECIMIENTOS

A toda la gente que de una u otra forma contribuyó en la obtención de la beca que me permitió realizar mi pasantía en IRCAM. En particular a mi familia, por todo su apoyo; a Pierre Donat-Bouillud, por su paciencia y disposición; y a todo el equipo MUTANT, por la acogida, simpatía y las interesantes conversaciones sobre música y computación.

ACKNOWLEDGMENTS

To all the people who helped, in one way or another, to obtain the scholarship that allowed me to accomplish my fellowship at IRCAM. Especially to my family, for all their support; to Pierre Donat-Bouillud, for his patience and disposition; and to the entire MUTANT team, for their acceptance, friendliness, and the interesting conversations about music and computer science.

GLOSARIO

INRIA: Institut National de Recherche en Informatique et en Automatique.
IRCAM: Institut de Recherche et Coordination Acoustique/Musique.
CNRS: Centre National de la Recherche Scientifique.
UDOO: Mini computador creado por Aidilab SRL SECO USA Inc. Incluye un procesador ARM Cortex A9, que corresponde a dos CPU Freescale i.MX 6. Además posee 1 GB de memoria RAM y una interfaz de Arduino.
Machine Listening: Técnica para obtener información útil desde señales de audio utilizando *software* y *hardware*.
Anthèmes 2: Es una pieza de música interactiva compuesta en 1997 por Pierre Boulez, fundador del IRCAM. Parte de los componentes electrónicos de la pieza son *samplers*, *frequency shifters*, armonizadores, *reverbs* y componentes de espacialización.
Grame: Centre National de Création Musicale, Lyon. France.
Repositorio Git: Espacio de almacenamiento del código fuente de programas escritos mediante el *software* de control de versiones Git.

GLOSSARY

INRIA: Institut National de Recherche en Informatique et en Automatique.
IRCAM: Institut de Recherche et Coordination Acoustique/Musique.
CNRS: Centre National de la Recherche Scientifique.
UDOO: Minicomputer created by Aidilab SRL SECO USA Inc. Includes an ARM Cortex A9 processor, which corresponds to two CPU Freescale i.MX 6. In addition, it has 1 GB RAM memory and an Arduino interface.
Machine Listening: Technique to obtain useful information from audio signals by using software and hardware.
Anthèmes 2: Interactive music piece composed in 1997 by Pierre Boulez, founder of IRCAM. Part of the electronic component of the piece are sampler, frequency shifters, harmonizers, reverbs and spacialization components.
Grame: Centre National de Création Musicale, Lyon. France.
Git Repository: Storage space of the source code of a program written by the software of control of Git version.

PRINCIPIO CIENTÍFICO

El principio científico utilizado en este artículo se basa en el aumento de la eficiencia en el uso de recursos computacionales y en la disminución del tiempo de ejecución de funciones utilizadas en el procesamiento de sonidos y efectos. El cambio en la arquitectura del procesamiento de señales de Antescofo, desde un *software* cuyas señales eran procesadas en *patches* externos por un sistema de paso de mensajes, hacia la integración de herramientas especializadas de procesamiento de señales, permite un aumento sustancial en la eficiencia en el tiempo de ejecución. El argumento principal de esto es que, al integrar por código herramientas modulares de procesamiento de señales, se permite una mejor utilización de los recursos computacionales. Esto se basa principalmente en la pre-

SCIENTIFIC PRINCIPLE

The scientific principle used in this article is based on the increase of efficiency in usage of computational power and the reduction of execution time in functions used by sound and effect processing. The change in Antescofo's signal processing architecture, from a software whose signals were processed in external patches by a message passing system, to the integration of tools specialized in signal processing, permits a substantial increase in execution time efficiency. The main arguments is, when modular signal processing tools are integrated by code, a better usage of computational powers occurs. This is based primarily in the pre-compilation of effects and signal processors at the time the score is loaded. With the old architecture, all the processing occurred outside

compilación de los efectos y procesadores de señales en el momento de cargar la partitura. Con la arquitectura anterior, todo el procesamiento ocurría fuera de Antescofo como una caja negra, en cambio, con las herramientas integradas en el código fuente del programa, se permite un mejor manejo de los recursos en tiempo de ejecución al tener acceso a optimización del código fuente de las herramientas modulares embebidas.

Antescofo as in a black box; however, with the tools integrated in the program's source code, it is possible to have a better control of execution times resources by having access to optimization of the source code of embedded modular tools.

REFERENCES

[1] GIAVITTO, J. L. et al. *Antescofo, a not-so-short introduction to version 0.x* [online]. 2015. Available at: http://support.ircam.fr/docs/Antescofo/AntescofoReference.pdf
[2] PUCKETTE, M. *Pd Documentation* [online]. Available at: http://puredata.info/docs/manuals/pd
[3] GAUDRAIN, E.; ORLAREY, Y. *A Faust Tutorial* [online]. Lyon, 2003. Available at: http://faust.grame.fr/images/faust-doc/Faust_tutorial.pdf
[4] FRIGO, M.; JOHNSON, S. *FFTW* [online]. 2012. Available at: http://www.fftw.org/fftw3.pdf
[5] DONAT-BOUILLUD, P. *Multimedia scheduling for interactive multimedia systems*. Multimedia [cs.MM]. 2015. Available at: https://hal.inria.fr/hal-01168098

EQUIPO DE INVESTIGADORES / RESEARCH TEAM


Nicolás Schmidt


Arshia Cont


Jean-Louis Giavitto