DEPARTAMENTO DE INFORMÁTICA

PROYECTO FIN DE CARRERA

APLICACIÓN DE EXPRESIVIDAD MUSICAL MEDIANTE UN SISTEMA EXPERTO BASADO EN REGLAS

Autor:
Jose María Gil Menéndez

Tutor:
Tomás de la Rosa Turbides

Leganés 8 de abril de 2015

Título: Aplicación de Expresividad Musical mediante un Sistema Experto Basado en Reglas
Autor: Jose María Gil Menéndez
Director: Tomás de la Rosa Turbides

EL TRIBUNAL

Presidente: Fernando Fernández Rebollo. Dpto. Informática. Área: Ciencias de la Computación e Inteligencia Artificial.

Secretaria: Nerea Luis Mingueza. Dpto Informática. Área: Ciencias de la Computación e Inteligencia Artificial.

Vocal: Iria Estévez Ayres. Dpto. Ingeniería Telemática.

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 23 de Abril de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Quiero dedicar el proyecto a todos los que han aportado y ayudado no solo durante este proyecto sino en mi vida.
Gracias a mis compañeros de universidad: Fran, Laura y Alberto, con quienes he compartido muy buenos momentos.
Gracias a Tomás y Sergio, por la oportunidad que me han dado para realizar este proyecto y sobre todo por su inestimable ayuda y enseñanza.
Gracias a Julio, por ser un referente, por su insistencia, confianza, ayuda y formación.
Gracias a mis padres, quienes han hecho posible que pueda mezclar música e informática a la vez, con su constante educación, plena dedicación y esfuerzo de todo tipo en nosotros.
A mis hermanos y Antonio, por estar y saber que siempre estarán apoyándome.
Y por último, a mi amor, aguantándome todas las noches frente al ordenador, y quién me hace ver todo mucho más fácil y saber qué es lo realmente importante en la vida.

¡MUCHAS GRACIAS A TODOS!

Resumen

La música es un arte, y un arte debe transmitir sentimientos. La expresividad es el medio por el cuál un músico muestra sus emociones al oyente, quién a su vez percibe las suyas propias. En resumen, la música sin expresividad no es más que un conjunto de sonidos que transcurren en el tiempo.
La mayoría de reproductores de melodías en formato *MIDI* presentan una melodía plana, sin expresividad, a no ser que el compositor defina mediante notación musical las variaciones en intensidad y volumen de la misma, siempre y cuando el programa sea capaz de reconocerlos. El objetivo principal del proyecto es crear un sistema experto en el cuál, a través de una entrada consistente en la composición de una melodía representada en formato *musicXML*, se apliquen reglas de expresividad musical de tal forma que la salida del sistema sea un archivo en formato *MIDI* en el que se distingan patrones que hagan al oyente sentir expresividad en la obra reproducida. Las composiciones pertenecerán a la época del Barroco y el instrumento elegido es el violín.
Palabras clave: Inteligencia Artificial, Música, Expresividad, Violín, Barroco, Sistema Experto, Reglas.

Abstract

Music is art, and all kind of art should make you feel different emotions. Musical expressiveness is the mean in which the players transmit their own ones to the listener.
Most of the *MIDI* players create a plain melody, with no expressiveness. Expressivenes is created by composers by defining musical symbols which modify both volume and intensity, as long as players recognize them.
The main goal for this project is to create an expert system that applies musical expressiveness to compositions. This goal is achieved reading a musical score in *musicXML* format and using expert rules that will contain expert knowledge about expressiveness for violin instrument in Baroque era. The output will be a *MIDI* file that reproduces the composition which has been modified according to the expressive rules. That is, the listener should feel emotions when listening the audio file.
Keywords: Artificial Intelligence, Music, Expresiveness, Violin, Baroque, Expert system, Rules.

Índice

Índice de figuras

Índice de cuadros

1. Introducción y objetivos

1.1. Introducción

Cualquier tipo de música, sea del estilo que sea, tiene como objetivo principal, comunicar sentimientos o sensaciones. Por lo tanto, es necesario que exista una expresividad en las melodías para que puedan ser interpretadas. Si toda melodía fuera plana, es decir, no tuviera variaciones, la música no tendría sentido puesto que no aportaría apenas más significado que el de las notas reproducidas. El significado de expresivo, según la definición de la R.A.E (Real Academia Española) es [1]:

"adj. Dicho de cualquier manifestación mímica, oral, escrita, musical o plástica: Que muestra con viveza los sentimientos de la persona que se manifiesta por aquellos medios."

Por tanto, la expresividad musical se puede entender como la capacidad de las personas para mostrar sus sentimientos mediante la música y que al mismo tiempo sean perceptibles por las demás personas.
Este trabajo se centra en la expresividad de las obras de música clásica pertenecientes a la época del Barroco. Se implementará un sistema experto capaz de (a partir de una partitura plana) interpretar una obra mediante una serie de patrones y modificarla de acuerdo a unas reglas establecidas producto de un conocimiento experto, de forma que se consiga percibir una expresividad que inicialmente no había.
Ahora bien, las interpretaciones que cada persona pueda hacer de una melodía posiblemente disten mucho las unas de las otras. Aún así, se trabajará con "convenios musicales" establecidos como portadores de cierta expresividad musical. En concreto, el sistema experto simulará la interpretación y expresión de un violín. Para lograrlo, el sistema experto será capaz de variar, por una parte, la dinámica de la obra, perteneciente a la intensidad y la velocidad de ésta, y por otra los efectos sonoros que simulen cierto instrumento, englobando tanto el tipo de sonido emitido como las diferentes formas de reproducirlo. Haciendo referencia a la dinámica de la obra, ésta es común a todos los instrumentos, por lo que se puede aplicar a cualquiera de ellos, generalmente de su misma familia o a cualquiera que pueda reproducir el mismo registro de notas. Por ejemplo, a un timbal no se le puede aplicar la misma dinámica que al violín, pero en cambio a un clarinete, o a un violonchelo sí puesto que la capacidad para emitir sonidos es la misma. En cuanto a las características propias del violín, los efectos a usar serán aquellos que aporten expresividad a la obra simulando la interpretación mediante este instrumento.

El desarrollo del trabajo estará marcado por las siguientes fases:

- Identificación de las reglas y patrones que establecerán el qué y el cómo modificar las distintas partes de una obra, a partir de ciertas condiciones que se cumplan en ésta.
- Identificación de los sistemas y tecnologías a utilizar para alcanzar el objetivo del trabajo.
- Establecer los distintos formatos digitales en los que estará representada la obra, para poder ser tratada por los distintos sistemas mencionados en el punto anterior.
- Implementación de los subsistemas que forman la arquitectura del sistema.

1.2. Objetivos

El objetivo principal del proyecto es producir una melodía en formato *MIDI* a partir de una partitura en formato *musicXML* donde se aprecie expresividad musical.
Para completar dicho objetivo, se realizarán la siguiente lista de subobjetivos:

- Subobjetivo 1: Generación de información en formato *musicXML*.
  El primer subobjetivo será crear o recopilar una base de datos con distintas melodías en dicho formato, bien mediante la adquisión de dichas partituras en servidores de internet o la creación de melodías que sirvan de pruebas para la experimentación del proyecto. Las obras estarán adaptadas al instrumento usado para el proyecto: el violín.

- Subobjetivo 2: Definición de reglas de expresividad mediante entrevista a un experto
  El segundo subobjetivo consiste en concertar una o varias entrevistas con un experto para poder extraer las reglas de expresividad para el violín y que serán usadas por el sistema.

- Subobjetivo 3: Creación de reglas de expresividad mediante un motor de reglas.
  El tercer subobjetivo consiste en la definición y codificación de reglas a partir del subobjetivo anterior. Para ello será necesaria la elección de un motor de reglas y la creación de una interfaz entre el diseño elegido y dicho motor de forma que los objetos representantes de la melodía mantengan su estructura y por tanto que la aplicación de expresividad sea transparente al proceso completo (esto es, que el tratamiento de la melodía no dependa de si a ésta se le ha aplicado la expresividad o no).

- Subobjetivo 4: Modificación de la información acorde a las reglas de expresividad encontradas.
  El cuarto subobjetivo consiste en, una vez obtenidas las reglas de expresividad, crear un mecanismo capaz de traducir dichas reglas para poder modificar la melodía codificada, y que servirá finalmente como salida del sistema experto. Esto se hará usando objetos *JAVA* que representen el elemento indivisible de la partitura: la nota. El conjunto de notas de la partitura se guardarán en una colección y se pasará al sistema experto, quién será el encargado de modificar las características de la nota: duración, volumen, altura, etc. Está modificación será la salida del sistema experto.

- Subobjetivo 5: Exportación de la información en formato *MIDI*.
  El quinto subobjetivo se basa en recoger la melodía ya modificada con las reglas de expresividad, entenderla y en función de los atributos de ésta escribir la melodía en formato *MIDI*. Se ha elegido este protocolo porque su API *JAVA* permite reprensentar todos los elementos necesarios para reconocer la expresividad de una forma sencilla y eficaz. El protocolo *MIDI* y sus mensajes son explicados en el apartado 2.2

- Subobjetivo 6: Evaluación de las melodias en formato *MIDI*.
  El sexto y último subobjetivo consistirá en evaluar de forma cualitativa si las melodías creadas por el sistema experto tienen una expresividad coherente o no.

Por tanto, teniendo en cuenta dicho objetivo y los correspondientes subobjetivos, la arquitectura del sistema se creará conforme a los mismos. La figura 16 representa la arquitectura del sistema.

1.3. Fases del proyecto

En esta sección se describirán las distintas fases del desarrollo necesarias para conseguir el objetivo del proyecto.

1.3.1. Fases de desarrollo

En la fase de desarrollo se han establecido los siguientes hitos:

- Estudio del formato *musicXML*.
- Desarrollo de un parser para la lectura de partituras en formato *musicXML*.
- Diseño y desarrollo de la representación de la melodía en *JAVA*.
- Adaptación del motor de reglas con el subsistema resultado del punto anterior.
- Procesamiento de la melodía en el motor de reglas y extracción de la expresividad en la misma.
- Conversión de la expresividad extraída por el motor de reglas para generar la salida en formato *MIDI*.

1.3.2. Fases de experimentación

Para cumplir con el objetivo del proyecto, se han redactado las siguientes fases de experimentación en el proyecto:

- Creación del conjunto de partituras sobre las que se aplicará la expresividad.
- Extracción de los patrones de expresividad de la melodía. Esto se implementará mediante las reglas definidas en la entrevista al experto y que serán las usadas por el sistema experto *JESS* para identificar dichos patrones.
- Evaluación de la salida generada. Dicha evaluación se realizará escuchando la melodía e identificando si el oyente es capaz de percibir la expresividad acorde a los patrones de la obra.

1.4. Estructura de la memoria

A modo de guía, a continuación se expone un resumen explicando brevemente cada uno de los capítulos en los que está estructurada esta memoria:

- Capitulo 1. Introducción y objetivos. Este capítulo está formado por tres secciones en las que se realiza una breve introducción sobre el proyecto, explicando los objetivos y subobjetivos del mismo así como las fases de desarrollo.
- Capitulo 2. Medios empleados. En este capítulo se establecen los conocimientos básicos de la teoría musical y también las tecnologías/herramientas usadas en el proyecto.
- Capitulo 3. Estado del arte. En este capítulo se exponen trabajos relacionados con el proyecto en el ámbito de la inteligencia artificial.

- Capitulo 4. Desarrollo. En este capítulo se define la arquitectura y flujo del sistema, detallando el desarrollo de cada una de las herramientas utilizadas en el proyecto y la relación entre éstas.
- Capitulo 5. Resultados. En este capítulo se expone la experimentación realizada y los resultados obtenidos.
- Capitulo 6. Conclusiones. En este capítulo se exponen las conclusiones extraídas de la realización del proyecto así como de su experimentación y resultados. También se incluyen aquí posibles líneas futuras del proyecto.
- Capitulo 7. Planificación. En este capítulo se explica la planificación seguida a lo largo del desarrollo del proyecto.
- Capitulo 8. Presupuesto. En este capítulo se explica el presupuesto necesario para la realización del proyecto.

2. Medios empleados

2.1. Teoría Musical

En esta sección se explican los elementos musicales que intervienen en el proyecto, de forma que se tengan unas nociones básicas de la teoría musical para poder comprender el funcionamiento del sistema.

2.1.1. Elementos básicos

- Melodía: Es una sucesión de sonidos y silencios, ordenados secuencialmente en el tiempo. Es un conjunto de sonidos con significado propio. [6]
- Nota: Es la unidad más básica de una melodía. Es decir, la melodía está formada por notas, cada una representando un sonido diferente (formado por una frecuencia determinada). La nomenclatura de éstas son: Do - Re - Mi - Fa - Sol - La - Si.
  Propiedades de las notas:
  - Altura o tono: Los sonidos poseen una determinada frecuencia, que determina su altura. De manera intuitiva, la frecuencia es la que indica si una nota es grave o aguda. Existe una relación en dicha frecuencia que hace que dos sonidos con diferente frecuencia se denominen con la misma nota (por ejemplo, Do). A esta relación se le denomina octava. La figura 1 muestra la nota Do con diferencia de una octava entre ellas:

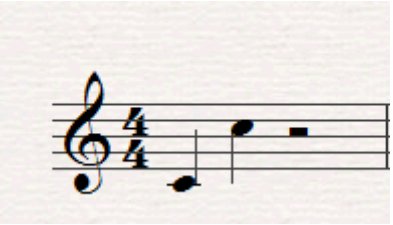

Figura 1: Altura entre notas.

    En este ejemplo, la segunda nota tendría mayor altura que la primera.

  - Intensidad: Esta propiedad establece si un sonido es fuerte o débil. Cabe destacar que esta propiedad es determinante en la expresividad de una melodía. Viene determinado por la amplitud de la onda que produce la nota. Su nomenclatura viene establecida por unas abreviaturas que indican la intensidad en ese momento, algunos ejemplos son:
    - Muy fuerte: ***ff***
    - Suave (Débil): ***p***

    Otra forma de variar la intensidad es mediante los reguladores, que van aumentando o disminuyendo la intensidad gradualmente desde un inicio hasta un final.
  - Timbre: Esta propiedad viene determinada por la forma de la onda del sonido y es la que hace que se diferencie que una cierta nota haya sido tocada, por ejemplo, por un violín o por un piano.

- Duración o figura: Cada nota es tocada durante cierto intervalo de tiempo. Este tiempo se suele representar mediante un pulso que se repite con cierta frecuencia (pulsos por minuto). Esta propiedad se representa mediante las figuras (ver Anexo 1, cuadro 17). Se exponen los siguientes ejemplos:
  - 1 pulso corresponde a la figura *Negra*, mostrada en la figura 2.

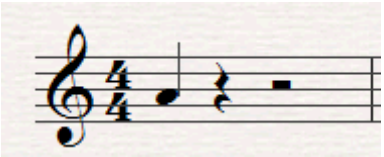

Figura 2: Duración correspondiente a una negra.

  - 2 pulsos corresponden a la figura *Blanca*, mostrada en la figura 3.

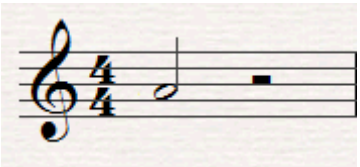

Figura 3: Duración correspondiente a una blanca.

    Al igual que la intensidad, la duración de las notas (el tiempo) es determinante en la expresividad de la melodía y puede modificarse en momentos concretos de la misma. Es decir, mediante una nomenclatura, el tiempo puede acelerarse o retardarse durante un fragmento de la melodía. A esto se le conoce como *accelerando* (acelerar) o *ritardando* (retardar).

- Alteración: Como se comentó anteriormente, existen siete tipos de notas. Sin embargo, se pueden representar cinco sonidos más, mediante las alteraciones: sostenidos (♯) y bemoles (♭). Éstos, se colocan en el lado izquierdo de la nota que se desea modificar. En términos de frecuencia de un sonido representado por una nota, los sostenidos y los bemoles la aumentan o disminuyen respectivamente. Es decir, los sostenidos hacen la nota más aguda y los bemoles más grave. La cantidad de frecuencia modificada corresponde a un semitono, que es la diferencia más pequeña que puede haber entre dos notas consecutivas.

- Silencio: Es un intervalo de tiempo en el que no se emite ningún sonido. Está representado por unos símbolos que equivalen a las figuras de las notas ya mencionadas. Estos símbolos, por tanto, son los que dan la duración al silencio. Por ejemplo:
  - El silencio correspondiente a 1 pulso se denomina *Silencio de negra*, mostrado en la figura 4.
  - El silencio correspondiente a 2 pulsos se denomina *Silencio de blanca*, mostrado en la figura 5.

Figura 4: Silencio de negra.

Figura 5: Silencio de blanca.

- Compás: Es el elemento que divide en partes iguales de tiempo una obra o melodía. Éste a su vez, está dividido en otras partes iguales, denominadas tiempos. Estos tiempos están representados por las figuras mencionadas anteriormente. Para determinar qué "cantidad" de tiempo debe albergar cada compás, éste tiene establecido al principio de la partitura una nomenclatura con dos números colocados verticalmente. El primero (empezando de abajo a arriba) indica el tipo de figura básica del compás (cada número está asociado a una figura, ver columna duración relativa del Anexo 1, cuadro 17) y el segundo indica la cantidad de ese tipo de figura que cabe en el compás. Por ejemplo, la figura 6 muestra que en un compás caben dos negras (el número 4 está asociado a la negra, y el número 2 indica la cantidad de notas de dicho tipo):

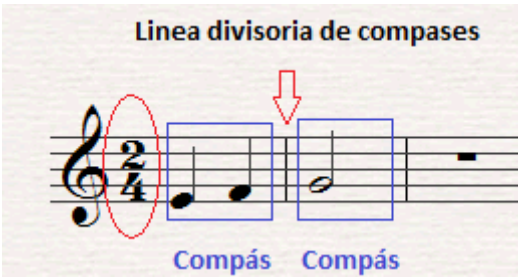

Figura 6: Descripción de compas.

2.1.2. Terminología de técnicas expresivas del violín

- Vibrato: Palabra de origen italiano que significa "vibrado". Es un efecto que produce una variación periódica en la frecuencia de la nota. Produce una sensación de vibración sobre la nota que se ejecuta, para ello la figura de ésta debe ser lo suficientemente larga como para poder aplicarlo.

- Glissando: Proviene de la palabra francesa *glisser* que significa *resbalar* o *deslizar*. Es un efecto que consiste en pasar de forma muy rápida de un sonido a otro, ejecutando todos los sonidos (no solo los representados por las notas) intermedios posibles entre la primera y última nota.

- Picato: Palabra italiana que significa *picado*. Es un efecto que acorta la duración de una nota, dando una sensación de "ligereza" a la melodía en la que se aplica.

- Dobles cuerdas: Es una ejecución de varias notas a la vez (acorde). Se realiza frotando más de una cuerda del violín a la vez. Es similar a pulsar varias teclas de un piano en el mismo instante.

2.1.3. Estructuras de la fraseología musical

A continuación se explicarán elementos más complejos sobre los que se aplicarán una serie de convenciones para crear expresividad a una melodía. Es decir, existen estructuras musicales que conforman patrones y que determinarán cómo y cuándo aplicar expresividad.
En primer lugar, para reconocer dichos patrones, la obra ha de analizarse de forma que se encuentren los siguientes elementos:

- Motivo: Es un conjunto de notas que forman una estructura musical la cual se repite varias veces a lo largo de la melodía. Normalmente está formado por uno o dos compases.
- Semifrase: Está formada por uno o más motivos. Normalmente son representadas mediante cadencias. Las cadencias son sucesiones de acordes, que dependiendo de la estructura, transmiten diferentes tipos de sensaciones al oyente. Se puede decir que, en las semifrases se encuentra el primer patrón sobre el que posteriormente se puede definir una expresividad musical.
- Frase: Está formada por dos o más semifrases. Las frases tienen sentido propio. Es decir, una frase terminará cuando la estructura que ésta tiene cambie totalmente con respecto a compases posteriores de la obra. Generalmente están marcadas mediante cadencias (contenidas en las semifrases) que las definen, aportando igualmente sensaciones al oyente. Por ejemplo, una frase puede aportar una situación de suspense (formada por una cadencia suspensiva) o una situación de fin (formada por una cadencia terminativa), como es el caso de los finales de las obras. La figura 7 muestra la estructura general de una frase.

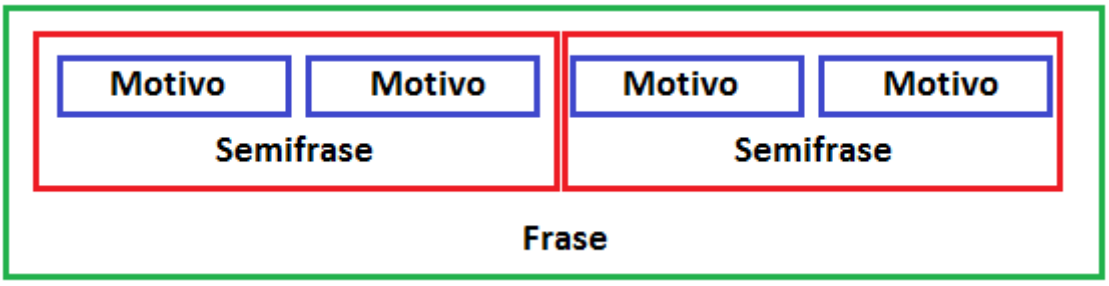

Figura 7: Estructura de una frase.

En una obra hay mucho más elementos a analizar, pero para el sistema experto solo son necesarios estos tres, de forma que una vez encontrados, pueda trabajar con ellos aplicando una serie de reglas que aporten expresividad a la obra. De forma resumida, el sistema experto encontrará las frases, y trabajará sobre los elementos que éstas contengan por separado. Como ejemplo para entender qué es y cómo identificar cada elemento, se expone un ejemplo de un fragmento de un vals de Mozart perteneciente al tercer Landler del KV 606 [2] y que es representado en la figura 8.

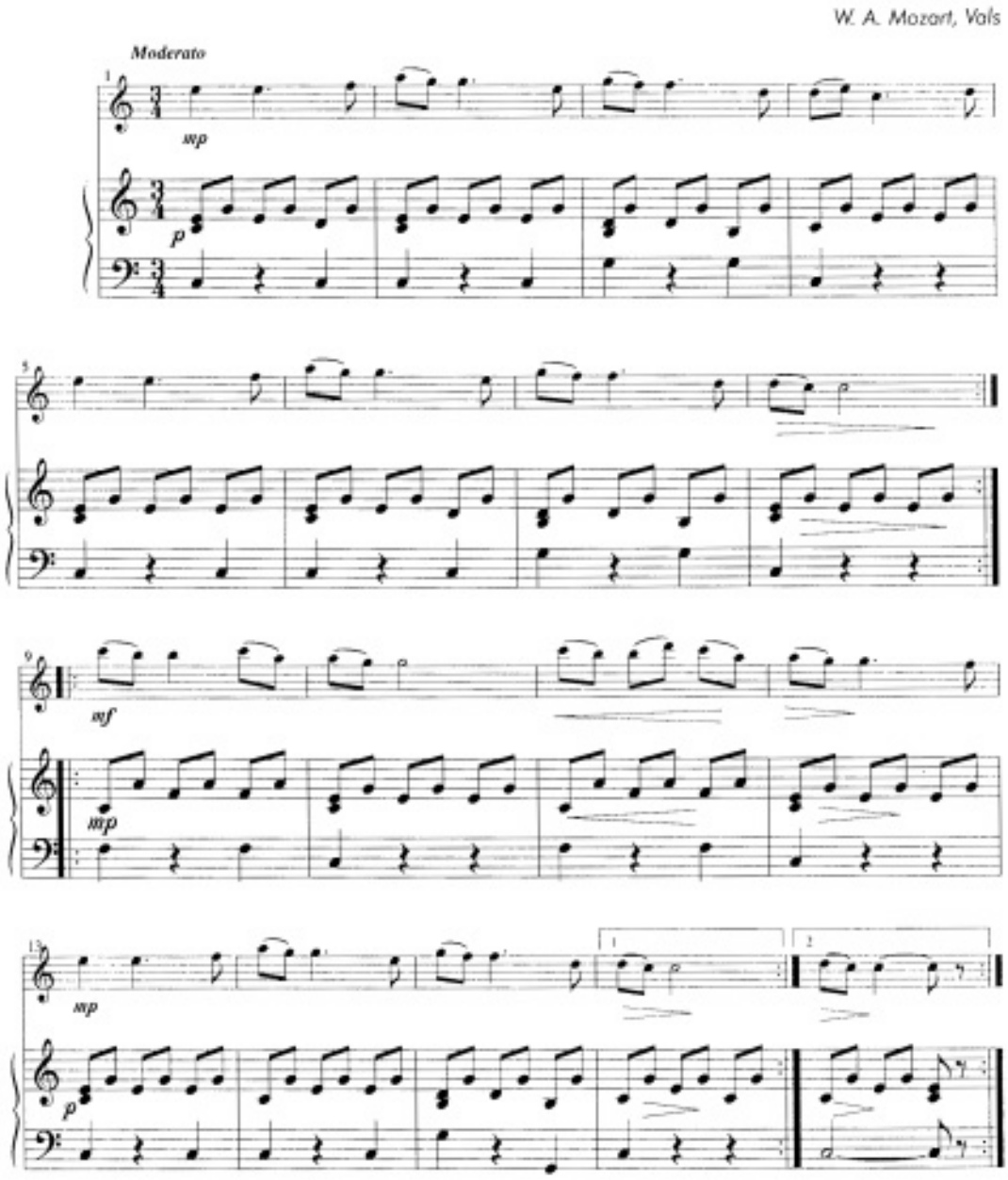

Figura 8: Fragmento de Vals de Mozart

La figura 9 muestra el análisis de dicho fragmento acorde a los elementos mencionados anteriormente.

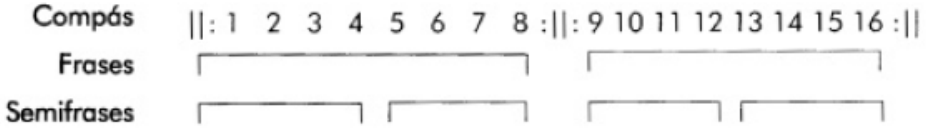

Figura 9: Análisis de la obra anterior

El fragmento se compone de dos frases, y éstas a su vez en dos semifrases cada una. En

este caso los motivos se pueden considerar cómo cada uno de los compases del fragmento.

2.2. MIDI

La tecnología *MIDI* (Musical Instrument Digital Interface) [29] es usada para sincronizar instrumentos electrónicos, ordenadores y hardware electrónico tales como secuenciadores o tarjetas de sonido entre otros. En vez de transmitir señales de audio digitales, envía mensajes sobre parámetros propios de la música, como por ejemplo, el tiempo, la duración y la intensidad, todos ellos explicados en la sección anterior. Aunque *MIDI* es usado como medio de comunicación entre distintos dispositivos, en este proyecto se usará como salida del sistema, en la que únicamente se realiza un mapeo de la melodía interpretada en objetos *JAVA* a dicha especificación, por lo que no se entrará en detalle en los términos relacionados con componentes hardware tales como voltajes, velocidad de transmisión, intensidad, conectores, etc. y sí en los dos principales componentes de este protocolo: el secuenciador y el sintetizador. Con respecto al sistema experto, en esta sección son explicados el protocolo *MIDI* y todos sus mensajes usados en este proyecto. Los principales mensajes que se utilizarán serán los correspondientes al tiempo e intensidad debido a que son los que definen en gran medida la expresividad musical de una obra. En la sección 4.5.3 se explica la conversión de estos mensajes al reproductor *MIDI* escrito en *JAVA* y su uso.
En primer lugar, se detallan los elementos estructurales del protocolo *MIDI* usados en el reproductor del sistema:

- Canal: Es el parámetro especificado por el cual se desea transmitir el mensaje. Hay un total de 16 canales. Cabe destacar que aunque en su programación dichos canales son enumerados del 0 al 15, al ser utilizados normalmente por músicos, éstos suelen mostrados del 1 al 16. Los canales sirven para mandar los mensajes de cada voz de la obra.

- Secuenciador: Es el dispositivo encargado de almacenar los datos *MIDI*, esto es, las notas, su velocidad, tipo de instrumento y efectos creados sobre la nota (por ejemplo, acentuarlas).

- Sintetizador: Es el dispositivo encargado de reproducir las notas, es decir, transformar los mensajes *MIDI* en sonido; el audio digital se representa mediante números y el sintetizador los reproduce en función de todos los mensajes almacenados en el secuenciador.

La figura 10 muestra la relación entre estos componentes en el sistema:

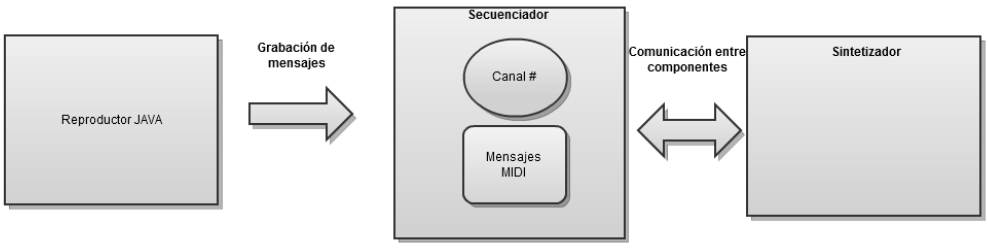

Figura 10: Comunicación entre secuenciador y sintetizador

Como se ha dicho anteriormente, el protocolo *MIDI* está basado en mensajes. La longitud de cada mensaje es al menos una cadena de 8 bits (1 byte). Este primer byte corresponde al estado del mensaje, es decir, el tipo de mensaje que servirá para identificar su función. Para ello, este byte es dividido en 2 *nibbles* de 4 bits. El primero especifica el tipo de mensaje y el segundo es el parámetro que determina su comportamiento. A continuación se describen los tipos de mensajes usados y su correspondiente *nibble*:

- *Note ON*: Indica el comienzo de una nota en el tiempo t especificado. Tiene un valor de 9.
- *Note OFF*: Indica el fin de una nota en el tiempo t especificado. Tiene un valor de 8. Por tanto, siendo t_1 el instante del mensaje en el que la nota comienza a sonar y siendo t_2 el instante en el que termina, la duración de la nota estará determinada por:

$$t_2 \text{ - } t_1$$

- *AfterTouch*: Este parámetro es usado para determinar la presión ejercida sobre una nota y es útil para representar los acentos creados para la expresividad en ciertas notas. Su valor es A.
- *Program Change*: Este parámetro cuyo valor es C, especifica el tipo de instrumento que se desea como salida. En nuestro caso, será el violín con un valor (en decimal) de 41.
- *Pitch Wheel*: Este parámetro cambia la frecuencia del sonido ejecutado, esto es, la altura de la nota. El cambio es pequeño y no corresponde a sonidos musicales por lo tanto se corresponde con una pequeña desafinación de la nota. Este parámetro es utilizado para simular el *vibrato* del violín. Su valor es E.
- *System common messages*: Estos mensajes del sistema son usados principalmente para sincronización entre canales (instrumentos) e inicialización del secuenciador y el sintetizador. En esta inicialización es especificado principalmente el *tempo* de la melodía.

Para explicar la función del segundo *nibble*, basta con poner un sencillo ejemplo: el mensaje *0x92* indica que comienza una nota en un determinado instante (9) en el canal número 2 (2).

2.3. MusicXML

MusicXML [5] es un formato basado en *XML* que se utiliza para representar la partitura de una obra, es decir, se centra en la notación musical de una melodía. Por tanto, mediante este formato es posible representar prácticamente todos los elementos musicales. Es posible escribir una partitura mediante *MusicXML* manualmente, sin embargo, debido al gran esfuerzo que esto supone en términos de control de errores (al fin y al cabo, se trata de un lenguaje con una gramática definida) y su larga definición para describir los diferentes elementos, existen programas que transforman la representación en notación musical de una partitura a este formato, como por ejemplo, Finale [3] o MuseScore [4].
Por otra parte, *MusicXML* posee sus propios DTDs y XSD [5], los cuales pueden ser distribuídos libremente bajo la licencia MusicXML Document Type Definition Public License [30].
La figura 11 muestra la equivalencia entre la nota musical *Sol* y su descripción en *MusicXML*.

```
<note>
  <pitch>
    <step>G</step>
    <octave>4</octave>
  </pitch>
  <duration>4</duration>
  <type>whole</type>
</note>
```

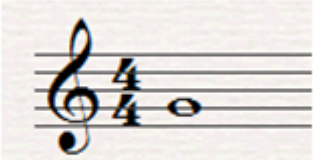

Figura 11: Traducción de una nota al lenguaje MusicXML

En dicha representación ya se puede observar algunos de los términos explicados. Por ejemplo, el *pitch* corresponde a la altura de la nota. En la etiqueta <step>, la letra G corresponde a la nota *Sol* (ver Anexo 1, cuadro 19) y la etiqueta *octave* corresponde al registro en el dominio de la frecuencia en el que se encuentra. Por otra parte, la etiqueta *duration* determina que la nota durará 4 pulsos y la etiqueta *type* establece la figura mediante la que se representa dicha duración, en este caso mediante el literal *whole*, que corresponde a la *redonda* (ver Anexo 1, cuadro 18). En la sección 4.4 se describe detalladamente cuál es el flujo del sistema para mapear estos elementos a objetos JAVA y por tanto obtener una representación de la partitura en *MusicXML* mediante los objetos correspondientes.

En la tabla 1 se exponen todos los elementos utilizados de la gramática de *MusicXML* junto a su descripción:

Parámetro	Valores	Nodo padre	Descripción
<score-partwise>	Ninguno	Ninguno	Es la raíz del documento *musicXML*
<part-list>	Ninguno	<score-partwise>	Contiene la cabecera del documento. Sus hijos definen las distintas voces (en nuestro caso, instrumentos) de la partitura.
<score-part>	Id:String	<part-list>	Representa cada instrumento (<score-instrument>) o voz de la partitura junto a su canal MIDI (<midi-instrument>).
<part>	Id:String	<score-partwise>	Definición de la melodía de cada instrumento o voz.
<measure>	number:int	<part>	Define cada compás del correspondiente instrumento o voz.
<note>	Ninguno	<measure>	Define cada una de las notas del compás
<pitch>	Ninguno	<note>	Define la altura de la nota
<type>	Ninguno	<note>	Define la duración de la nota
<step>	Ninguno	<pitch>	Define la nota (ver Anexo 1, cuadro 18)
<octave>	Ninguno	<pitch>	Define la octava en la que se encuentra la nota.
<alter>	Ninguno	<pitch>	Define si la nota posee un sostenido o bemol (o ninguno).

Cuadro 1: Descripción de los parámetros de MusicXML

Finalmente, la figura 12 muestra un ejemplo conteniendo todos estos elementos, y su correspondiente notación musical.

```
<score-partwise>
  <part-list>
    <score-part id="P1">
      <part-name>Violin</part-name>
      <score-instrument id="P1-I3">
        <instrument-name>Violin</instrument-name>
        </score-instrument>
      <midi-instrument id="P1-I3">
        <midi-channel>1</midi-channel>
        <midi-program>41</midi-program>
        </midi-instrument>
      </score-part>
    </part-list>
  <part id="P1">
    <measure number="1">
        <pitch>
          <step>C</step>
          <octave>5</octave>
          </pitch>
        <duration>4</duration>
        <voice>1</voice>
        <type>whole</type>
        </note>
      </measure>
    <measure number="2">
      <note>
        <rest/>
        <duration>4</duration>
        <voice>1</voice>
        </note>
      <barline location="right">
        <bar-style>light-heavy</bar-style>
        </barline>
      </measure>
    </part>
  </score-partwise>
```

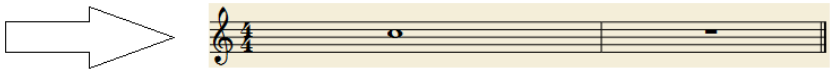

Figura 12: Obra sencilla escrita en musicXML

En ella vemos en primer lugar el nombre del instrumento: "Violin" y su correspondiente canal MIDI, el 1. Asimismo se puede observar que el atributo <midi-program> está relacionado con el atributo *program change* del protocolo MIDI. Después, cabe destacar que hay dos compases, con una nota cada uno. La primera, un *Do* situado en la quinta octava (altura) con una duración de 4 tiempos de compás, es decir, una redonda (véase que corresponde con el atributo <type>: *whole*). La segunda nota es un silencio de redonda, representada por el símbolo <rest/> y el valor del atributo <duration>.

2.4. Java

Para el desarrollo del trabajo, es necesario implementar un parser que transforme la partitura en formato *musicXML* para su posterior tratamiento. Por otra parte, se implementará un reproductor de música *MIDI*. Para ello, se usarán las siguientes librerías de *JAVA*:

- JDOM [31]: Es un API para poder leer documentos en formato *XML* de forma muy cómoda e intuitiva. Está compuesto por 3 paquetes principales:
  - Org.jdom: Representará principalmente el documento de entrada, en nuestro caso una partitura en formato *musicXML*. La clase Document representará este documento, y la clase Attribute representará los atributos que este tenga.
  - Org.jdom.input: Contiene las clases constructoras para crear los documentos *XML*.
  - Org.jdom.output: Contiene las clases que se usarán para crear la salida de la clase Document creada. A continuación se muestra un ejemplo de cómo se adapta el formato *XML* al código *JAVA*:

- Por ejemplo, si se desea obtener la duración de la nota representada en el apartado anterior:

  <duration>4</duration>

  La librería *JDOM* permitiría obtenerla fácilmente con la siguiente función:

  String duracion = duration.getText()

- JAVA SOUND [32]: Es una API que permite capturar, procesar y reproducir música *MIDI*. La tabla 2 muestra la lista con las principales clases e interfaces que se usarán en el tratamiento de música *MIDI*.

Interfaces	Clases
ControllerEventListener	Instrument
MetaEventListener	MetaMessage
MidiChannel	MidiDevice
MidiDevice	MidiEvent
Receiver	MidiFileFormat
Sequencer	Midimessage

Cuadro 2: Interfaces y clases de Java Sound

El uso de estos paquete se divide en dos clases principales:

- Instrumento.java: Se encargará de generar los mensajes *MIDI* y generar las *Tracks* de la secuencia. En esta clase se utilizarán los siguientes paquetes:
  javax.sound.midi.InvalidMidiDataException;
  javax.sound.midi.MidiEvent;
  javax.sound.midi.ShortMessage;
  javax.sound.midi.Track;

  La clase ShortMessage se encargará de crear el mensaje (el tipo de mensaje, la longitud de los datos del mensaje y los datos de éste)
  La clase MidiEvent utilizará este mensaje para crear un evento *MIDI*.
  La clase Track se encargará de añadir dicho evento a la secuencia *MIDI*.
  Por último, la clase InvalidMidiDataException contiene la excepción que indica, en caso de que ocurra, algún error en la composición del mensaje.

- Reproductor.java
  javax.sound.midi.InvalidMidiDataException;
  javax.sound.midi.MetaEventListener;
  javax.sound.midi.MetaMessage;
  javax.sound.midi.MidiEvent;
  javax.sound.midi.MidiUnavailableException;
  javax.sound.midi.Sequence;
  javax.sound.midi.Sequencer;
  javax.sound.midi.Track;

Esta clase creará y contendrá las secuencias *MIDI* creadas por Instrumento.java y se encargará de escribir éstas en el fichero de salida.
La clase Sequence contendrá toda la información musical (encapsuladas en la clase Track) y también la información temporal (velocidad del pulso de la secuencia) de éstas. Esta información es la que define la velocidad con la que se reproduce la obra.
La clase Sequencer contendrá un objeto de la clase anterior y será el encargado de escribir la obra en formato *MIDI*.
La clase MetaMessage contendrá la metadata de los mensajes, esto es: eventos tales como el *tempo*, la clave de la obra, el instrumento, etc. Finalmenta la interfaz MetaEventListener representa un *listener* cuyos eventos son los MidiEvent's. La finalidad de esta interfaz es poder reproducir la obra en tiempo real en lugar de escribirla. Por tanto, la clase Reproductor.java implementará dicha interfaz.

El siguiente diagrama de clases expone todo lo mencionado anteriormente:

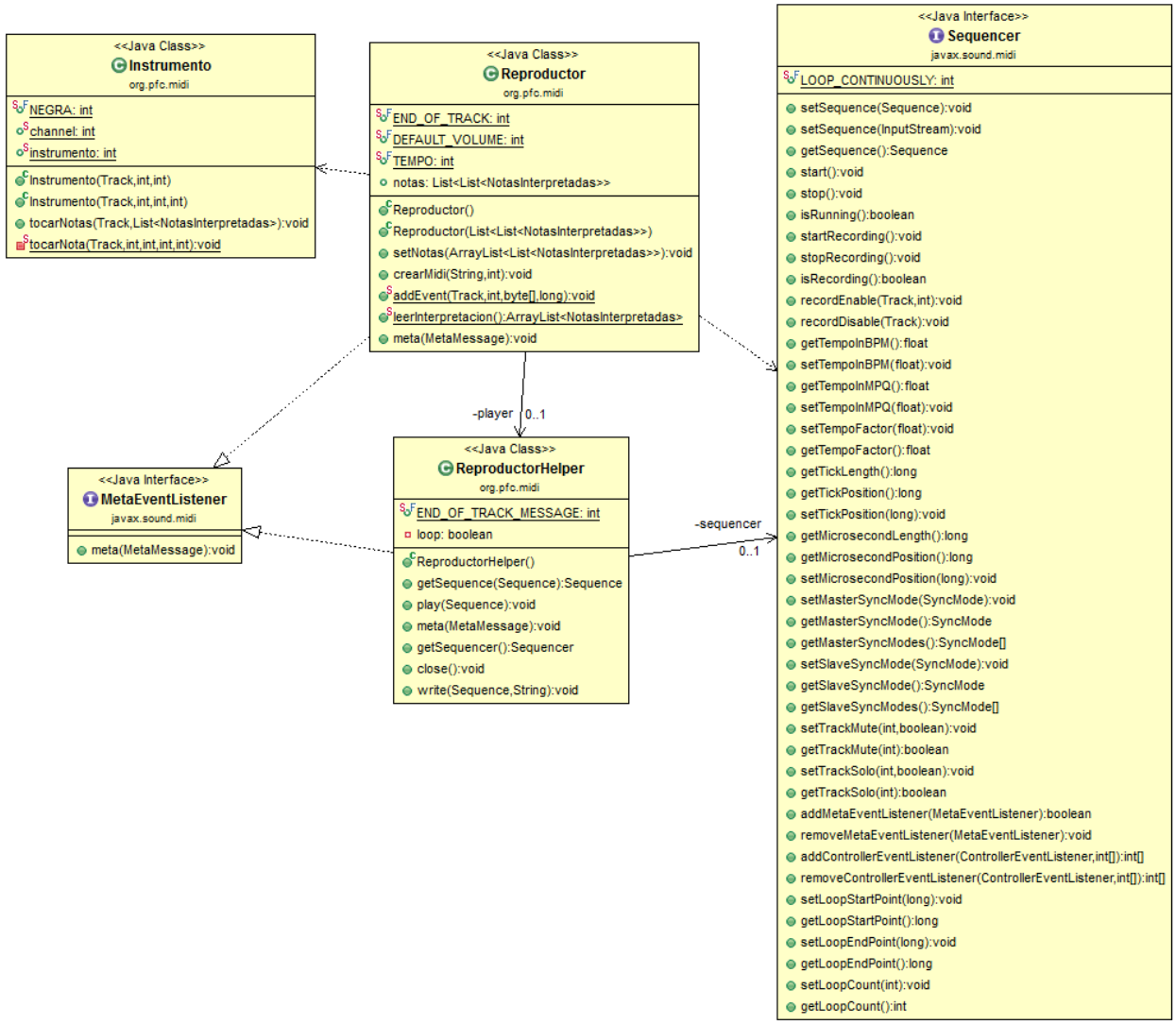

Figura 13: Diagrama de clases UML para la implementación MIDI

2.5. Jess

Jess [22] en un motor de reglas escrito íntegramente en *JAVA*, lo que hace trivial la integración de este medio con el resto del sistema.
Su programación se lleva a cabo mediante la definición de reglas y está basado en *CLIPS* [23], uno de los lenguajes de programación más usados para sistemas expertos. Según sus autores, *JESS* está orientado a proporcionar a las aplicaciones *JAVA* la capacidad de razonar [24].
A continuación se explican los aspectos más importantes de *JESS* acorde su uso en el proyecto:

- Reglas: Definición y Algoritmo.
  En las secciones 4.3 y 4.4 se explican detalladamente las reglas programadas para el sistema, aquí se describe cual es la definición y funcionamiento de este motor de reglas: El motor de reglas de *JESS* está basado en el llamado algoritmo *RETE* [33]. Este algoritmo fue creado por el Dr. Charles L. Forgy en la Carnegie Mellon University.

Surge como motivación a la siguiente ineficiencia computacional a la hora de evaluar un conjunto de hechos (en nuestro caso estos hechos serían las frases, semifrases, motivos, etc) dado un conjunto de reglas (en nuestro caso estas reglas serían el conocimiento experto descrito en la sección 4.3):

Por ejemplo, dada una regla como ésta:

```
(defrule disminuir_motivo_repetido
    (declare (salience 2))
    ?m1 <- (motivo (id ?id1) (semifrase ?sf1) (frase ?f) (notas $?notas1) (base ?base1))
    ?m2 <- (motivo (id ?id2) (semifrase ?sf2) (frase ?f) (notas $?notas2) (base ?base2))
    (test (= 0 (str-compare(implode$ $?notas1)(implode$ $?notas2))))
    (test (> ?sf2 ?sf1))
=>
    (modify ?m2 (base (- ?base2 10)))
    (assert (reglaAplicada (nombre disminuir_motivo_repetido) (motivo ?id2))))
```

Figura 14: Ejemplo de regla del sistema experto

Donde la traducción a pseudocódigo podría ser:

Regla Disminuir-motivo-repetido:
Si un motivo ?m1 dados sus atributos
Y otro motivo ?m2 dados sus atributos
Si las notas ?notas1 del motivo ?m1 son iguales a las notas ?notas2 del motivo ?m2
Y si la semifrase ?sf2 del motivo ?m2 es mayor que la semifrase ?sf1 del motivo ?m1
Entonces:
Modifica la base de volumen ?base2 del motivo ?m2 y resta 10 a su valor actual
inserta el hecho llamado *reglaAplicada* con el atributo nombre con valor
"disminuir_motivo_repetido" y el atributo motivo el valor de ?id2.

Así, en este caso, *motivo*, *semifrase*, *frase* y *reglaAplicada* serían los hechos.
En *JESS*, cada hecho tiene una plantilla. Esta plantilla ha de tener un nombre y un conjunto de *slots*. Los *slots* se pueden entender como las columnas de una tabla de una base de datos relacional o como las propiedades de JavaBean (POJOs) [25].
Todas las reglas implementadas serán fijas mientras que lo que cambia son los hechos, que se encuentran en la memoria del programa (por tanto, ésta estaría cambiando continuamente). Sin embargo, se observó empíricamente que esta memoria se mantiene igual durante la mayor parte del tiempo de ejecución, es decir, que el porcentaje de cambio de cada hecho por unidad de tiempo es muy pequeña. Dado el pseudo-código anterior, la implementación más sencilla sería evaluar cada una de las reglas entre todos los hechos (en inglés: Left Hand Side) y aplicar las funciones en caso de cumplirlas (en inglés: Right Hand Side). Así, la implementación básica está basada en "reglas que encuentran hechos" y su complejidad es $O(RF\hat{P})$ donde R es el conjunto de reglas, F el cónjunto de hechos y P es la media del número de patrones especificados por cada regla en la parte LHS (Left Hand Side).
Por tanto, la solución empleada por el algoritmo RETE es recordar las evaluaciones realizadas en los ciclos anteriores: sólo los nuevos hechos generados en una iteración

serán otra vez evaluados por la parte LHS de las reglas. Además, no solo será este conjunto de hechos el nuevo objetivo de las reglas sino que solo un subconjunto de reglas (las que prueben ser más relevantes) serán las asignadas para tal evaluación. Mediante esta solución, la complejidad pasa a ser O(RFP).
A continuación se resume cómo trabaja *JESS* con el motor de reglas y hechos. La figura 15 muestra el esquema básico de un sistema de producción [40]:

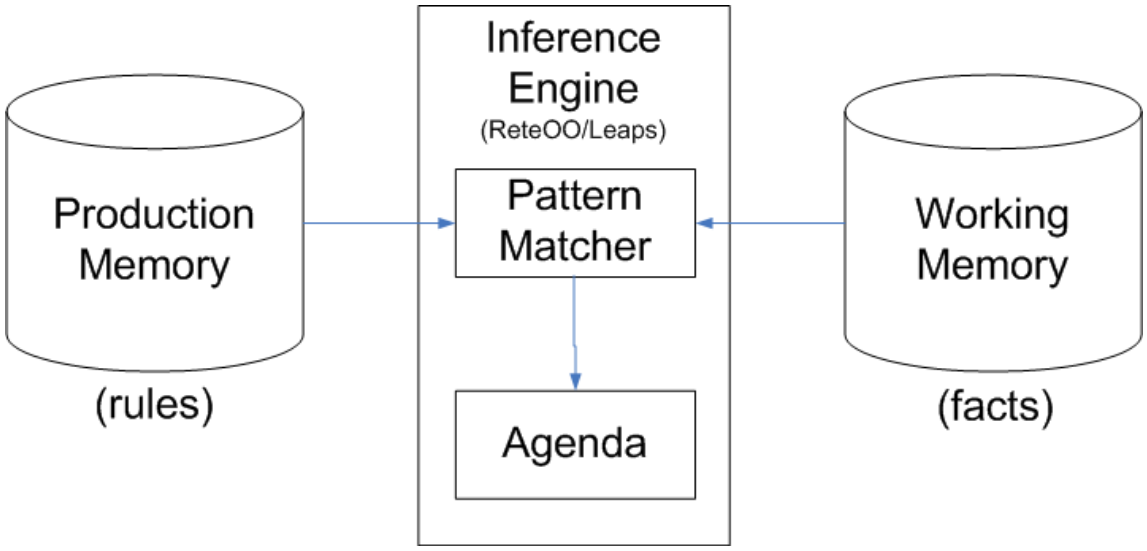

Figura 15: Esquema del algoritmo RETE

Así, las reglas son almacenadas en la zona "Production Memory" y los hechos encontrado mediante el motor de inferencias son almacenados en la zona "Working Memory" y es aquí donde éstos pueden ser modificados o eliminados. Por otra parte, la Agenda es la que resuelve los conflictos entre las distintas reglas que son verdaderas para el mismo hecho.

3. Estado del arte

En primer lugar se hace una breve introducción a la codificación de la música y por qué es posible aplicar inteligencia artificial sobre ésta. Ambos campos llevan unidos ya varios años, atendiendo a la siguiente definición [7]:

"La música es el arte de organizar sensible y lógicamente una combinación coherente de sonidos y silencios utilizando los principios fundamentales de la melodía, la armonía y el ritmo, mediante la intervención de complejos procesos psico-anímicos"

Se puede decir que la relación entre Inteligencia Artificial y la música se encuentra desde que el sonido ha podido ser codificado y grabado de múltiples formas, dividiéndose principalmente en dos [12]:

- Grabación analógica del sonido: Es el almacenamiento en aquellos medios en los que se almacenaba las señales analógicas de los sonidos. Algunos ejemplos son los vinilos, discos de acetato, magnetófono, etc. En esta sección no se hará hincapié sobre este procedimiento pero hay una gran variedad de documentación en internet que lo explica detalladamente.

- Grabación digital del sonido: Es el almacenamiento de una señal digital resultado de una conversión de la señal analógica. Este procedimiento se puede decir que es el que hace posible que la música sea representada, de múltiples formas, en un ordenador.

Por tanto, en primer lugar, para poder representar el sonido, hace falta captar su señal analógica y ser capaz de representar digitalmente dicha señal. Para ello, se utilizan dos procesos: el muestreo y la cuantificación digital de la señal eléctrica que representa una onda sonora [11].
Una vez obtenido el audio digital, surge la necesidad de almacenar dicha información en ficheros o soportes físicos, dando lugar a numerosos formatos para representar el audio digital tales como *mp3, wav, aa3, .ogg* ó *wma*. Cabe destacar que el formato *MIDI* no es un formato de audio digital sino un protocolo de comunicación entre distintos dispositivos musicales electrónicos.
Por tanto, conseguir representar digitalmente la música significa poseer datos y poseer datos significa poder crear y aplicar técnicas que simulen el razonamiento humano sobre ésta, naciendo así la investigación de la inteligencia artificial sobre la música.

3.1. La inteligencia artificial y la música

La aplicación de la inteligencia artificial sobre la música se ha llevado acabo desde hace varios años. El primer trabajo aparece en 1980, cuando R.Kh.Zaripov publicó el primer artículo sobre composición musical usando un ordenador llamado "Ural-1" [13].
Los sistemas de inteligencia artificial se pueden clasificar en 4 principales categorías:

- Sistemas de composición musical: Son aquellos que generan melodías armónicamente correctas, es decir, no basta con generar sonidos si no que éstos sean agradables al oído humano. En este campo se han hecho grandes avances como por ejemplo la investigación llevada a cabo por David Cope [14], quién ha creado un software llamado *EMMY* que se encarga de simular obras con patrones similares a otros compositores como por ejemplo Mozart, Bach o Chopin.

- Sistemas de improvisación musical: Son aquellos que generan melodías a partir de bases musicales. Estos sistemas normalmente se basan en un patrón musical que va siendo modificado en el tiempo, creando variaciones del mismo. También existen estudios en los cuales la improvisación musical se realiza sin ningún tipo de configuración basada en reglas, consiguiendo generar melodías que no pertenezcan a ningún tipo de contribución humana (las bases musicales) pero que simule una improvisación coherente (tal y como lo hiciera un humano) [15]

- Sistema de cognición humana: Son aquellos sistemas que tratan de encontrar el comportamiento de una melodía en la psicología humana [8]. Estos sistemas son ahora mismo la tecnología *"cutting-edge"* debido a la aparición de programas tales como iTunes, Spotify, Napster, etc. Estos programas usan la inteligencia artificial en la música para extraer estados de ánimo transmitidas por las canciones, categorizándolas en grupos y sirviendo así para crear "recomendaciones" al usuario con el objetivo de ampliar el catálogo musical.

- Sistemas de interpretación: Son aquellos sistemas que se encargan de aportar expresividad a las melodías, es decir, no generar simplemente una melodía plana en la que solo se aprecien sonidos sino que también sea capaz de provocar sentimientos al oyente.

Este proyecto se enmarcaría dentro de la cuarta categoría. Es por ello que se dedica una sección especial mostrando el estado del arte en los sistemas de este tipo.

3.2. Sistemas de expresividad musical

Uno de los primeros sistemas expertos enfocados a la expresividad musical fue creado por Margaret L. Johnson ([17]). Este sistema experto se basa en las *fugas* de "El clave bien temperado" de J.S Bach [19]. En primer lugar, extrae las reglas de dos expertos en dichas *fugas*. Tras codificar las reglas en el sistema experto, la salida de consiste en un conjunto de instrucciones en la duración y articulación de las notas. En las conclusiones del trabajo se puede leer como dichas instrucciones son similares a ediciones conocidas de "El clave bien temperado" [16].
Otro sistema encargado de crear expresividad musical fue el software *DIRECTOR MUSICES* [20]. Se trata de un sistema basado en reglas para el tiempo, articulación y dinámica (volumen) de la obra, produciendo una salida *MIDI*. Las reglas se categorizan en tres clases:

- Diferencias entre los tonos: Busca las diferencias entre los tonos, como por ejemplo cambios en la duración, en la altura de éstos, etc.

- Similitud entre los tonos: Son aquellas que marcan los límites entre las distintas frases de la obra.

- Agrupación de las distintas voces: Relaciona las distintas voces en cada momento de la melodía.

El trabajo realizado por Canazza et al. (1999) [21] consiste en un sistema capaz de modificar la expresividad en tiempo real. El modelo calcula las desviaciones de los parámetros musicales considerados relevantes en la expresividad y la plasma mediante un motor de procesamiento de sonido.

En 2007, Josep Lluis Arcos desarrolla un proyecto llamado *SAXEX* [41] que consiste en un sistema de razonamiento basado en casos capaz de generar expresividad basándose en ejecuciones musicales hechas por humano. Se divide básicamente en dos mecanismos básicos:

- Almacenamiento de problemas resueltos o casos de uso, todos ellos agrupados mediante algún criterio de similitud. Los casos de uso aquí consistirían en grabaciones de obras ejecutados por músicos.
- La aplicación de la solución a un nuevo caso de uso teniendo como referencia aquellas usadas en casos de usos anteriores.

Para la extracción de expresividad, *SAXEX* usa un modelo llamado *SMS* (Spectral Modelling Synthesis), útil no solo para dicha extracción sino también para la transformación a una melodía usando como base la original [42].
Las melodías en las que se centró originalmente este sistema están enfocadas al saxofón y al Jazz, con distintos grados de expresividad, incluyendo siempre una versión inexpresiva de cada una de ellas. Así, el conjunto de casos sería la aplicación del modelo SMS para la extración de elementos expresivos más las partituras de Jazz usadas.
Por tanto, dado este conjunto de casos, la entrada del programa es una nueva melodía de Jazz a la cual se le infiere un posible conjunto de aplicación expresivas. Una vez la expresividad es aplicada, se vuelve a usar el modelo SMS para transformar la melodía acorde a las modificaciones hechas por el método de razonamiento basado en casos.
Por último, en 2013, Alfonso Benetti Jr. publicó un estudio sobre la expresividad musical y la extracción de sus características basado en entrevistas a 20 pianistas y en el que definía su investigación basado en las siguientes cuestiones [18]:

- ¿Qué estrategias son aplicadas realmente por los pianistas?: Los entrevistados definieron cuatro técnicas principales: articulación, pedal, fraseo y *rubato*.
- ¿Cómo organizar las reglas y decisiones para crear una correcta expresividad musical?: Dadas las técnicas anteriores, el trabajo concluye que el sistema experto ha de ordenar las reglas y decisiones en el siguiente orden: articulación, *rubato*, pedal, y fraseo.
- ¿Cómo definen los pianistas la "expresividad"?: Según el 60 % , la expresividad es la herramienta de comunicación entre el músico y el oyente. El resto asocian la expresividad al espacio de libertad del músico para recrear sus sentimientos, a fenómenos culturales y a la "músicalidad" de éste (imaginación, intuición, sus sensaciones, etc).

4. Desarrollo

4.1. Requisitos de Usuario

En esta sección de describen los requisitos principales para el desarrollo del proyecto. Se muestra una tabla resumen de los requisitos de usuario que describen el sistema categorizados por tipo de requisito, identificador, nombre y descripción. Todos los requisitos serán considerados como esenciales y con prioridad alta.

Tipo de Requisito	Identificador	Nombre	Descripción
Capacidad	RUC-01	Lectura de fichero *musicXML*	La aplicación permitirá leer ficheros *XML* cumpliendo el formato DTD MusicXML 2.0 Partwise y XSD MusicXML
Capacidad	RUC-02	Selección de fichero de entrada	La aplicación permitirá seleccionar el fichero de entrada que contiene la partitura.
Capacidad	RUC-03	Selección de voces de la obra	La aplicación permitirá seleccionar el número de voces de la obra que se quieren analizar
Capacidad	RUC-04	Aplicación de expresividad automática	La aplicación permitirá definir si aplicar expresividad automáticamente
Capacidad	RUC-05	Aplicación de expresividad manual	La aplicación permitirá definir si aplicar expresividad manualmente
Capacidad	RUC-06	Representación de la melodía en *MIDI*	La aplicación representará la melodía en el protocolo *MIDI* de acuerdo al estandar "The Standard MIDI Files (SMF) Specification [RP-001]"
Capacidad	RUC-07	Exportación a fichero de la melodía *MIDI*	La aplicación permitirá elegir el fichero en el que exportar la melodía en formato *MIDI*
Capacidad	RUC-08	Reproducir de la melodía *MIDI*	La aplicación permitirá seleccionar si la melodía ha de ser reproducida tras aplicar la expresividad.
Resticción	RUR-01	Obra para violín *musicXML*	La aplicación no permitirá leer obras cuyo instrumento *MIDI* definido no sea el violín.
Resticción	RUR-02	Voces de la obra *musicXML*	La aplicación no permitirá seleccionar más voces que las que contiene la obra.
Resticción	RUR-03	Sistema Operativo *Windows*	La aplicación deberá ser ejecutada en Windows, desde la versión XP hasta la versión 8.
Resticción	RUR-04	*JAVA JDK*	El sistema operativo deberá tener el paquete JAVA JDK instalado al menos desde la versión 1.6
Resticción	RUR-05	Librería *JESS*	El sistema operativo deberá tener la librería JESS instalada, todas las versiones son compatibles hasta la fecha.

Cuadro 3: Tabla de requisitos de usuario de capacidad y restricción

4.2. Arquitectura del sistema

En esta sección se exponen los distintos módulos del sistema, así como la entrada y salida tanto del sistema en general como de cada uno de los módulos. La figura 16 muestra la comunicación entre los diferentes módulos de la arquitectura.

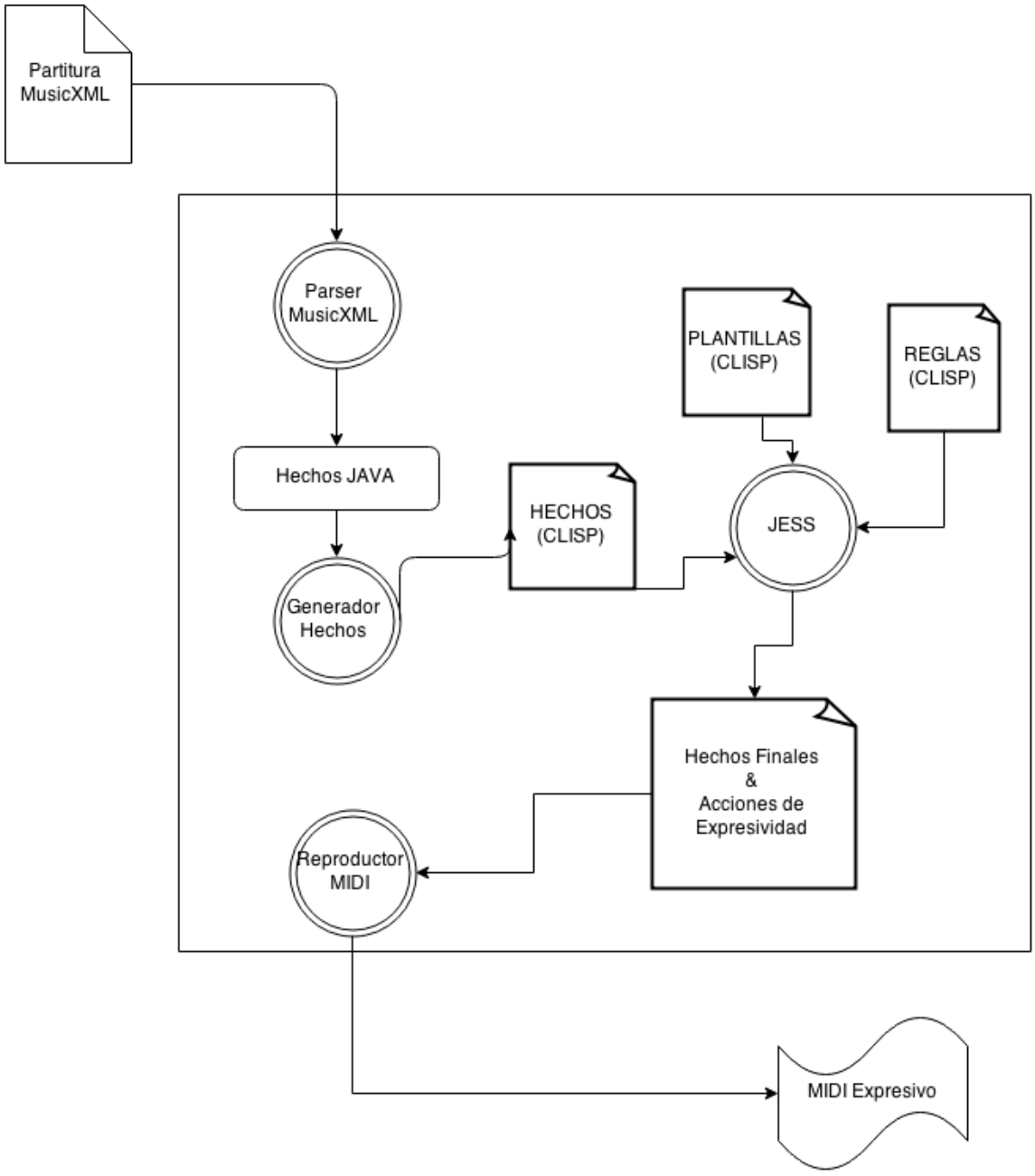

Figura 16: Arquitectura general del sistema

4.2.1. Parser de partituras XML

- Entrada: Archivo en formato *musicXML* (Partitura)
- Salida: Representación en objetos *JAVA* de la partitura.

La funcionalidad de este parser (*JAVA*) es la de transformar la partitura de entrada a una representación en memoria de todos los elementos necesarios: frases, semifrases, motivo y notas de ésta. Esta representación será la comunicación con el siguiente módulo para poder crear un fichero reconocible por *CLIPS*, para su posterior tratamiento mediante las reglas definidas para el sistema experto. Este fichero sería el estado inicial.

- Se parsea el formato *musicXML* a una gramática en *JAVA*. Esta gramática representa los componentes propios de una partitura. Estas son: la clave, las distintas voces, el compás, el tempo, los compases, las figuras y las notas.

  A continuación se muestra un ejemplo del proceso, usando como refencia la partitura mostrada en la figura 8:

  1. Dado el primer compás de la obra en formato *musicXML*:

```
<part id="P1">
    <measure number="1">
        <attributes>
            <divisions>2</divisions>
            <key>
                <fifths>0</fifths>
            </key>
            <time>
                <beats>3</beats>
                <beat-type>4</beat-type>
            </time>
            <clef>
                <sign>G</sign>
                <line>2</line>
            </clef>
        </attributes>
        <note>
            <pitch>
                <step>E</step>
                <octave>5</octave>
            </pitch>
            <duration>2</duration>
            <voice>1</voice>
            <type>quarter</type>
            <stem>down</stem>
        </note>
        <note>
            <pitch>
                <step>E</step>
                <octave>5</octave>
            </pitch>
            <duration>3</duration>
            <voice>1</voice>
            <type>quarter</type>
            <dot/>
            <stem>down</stem>
        </note>
        <note>
            <pitch>
                <step>F</step>
                <octave>5</octave>
            </pitch>
            <duration>1</duration>
            <voice>1</voice>
            <type>eighth</type>
            <stem>down</stem>
        </note>
    </measure>
```

Figura 17: Compás en formato musicXML

El sistema extraerá, en primer lugar, el compás y la clave. Después, se crea la voz 1, empezando por el compás 1 y las 3 notas pertenecientes a éste, y así sucesivamente

con los demás compases.

- A partir de la creación de objetos, se buscan las frases, semifrases y motivos de la obra. Esta parte del sistema extrae las frases, semifrases y motivos de la obra automáticamente o manualmente. La parte manual consiste en ejecutar el programa haciendo referencia a un fichero de metadata conteniendo dicha información (ver Manual de usuario). Si se ejecuta eligiendo la forma automática, el sistema interpreta que la mitad de la obra es una frase, y la segunda mitad otra. Después, cada frase estaría formada por dos semifrases y cada semifrase estaría compuesta por dos motivos.
- Por último, se guarda en memoria la representación de todos estos elementos para posteriormente ser la entrada del generador de hechos del sistema experto.

La arquitectura de este módulo queda representada en la figura 18

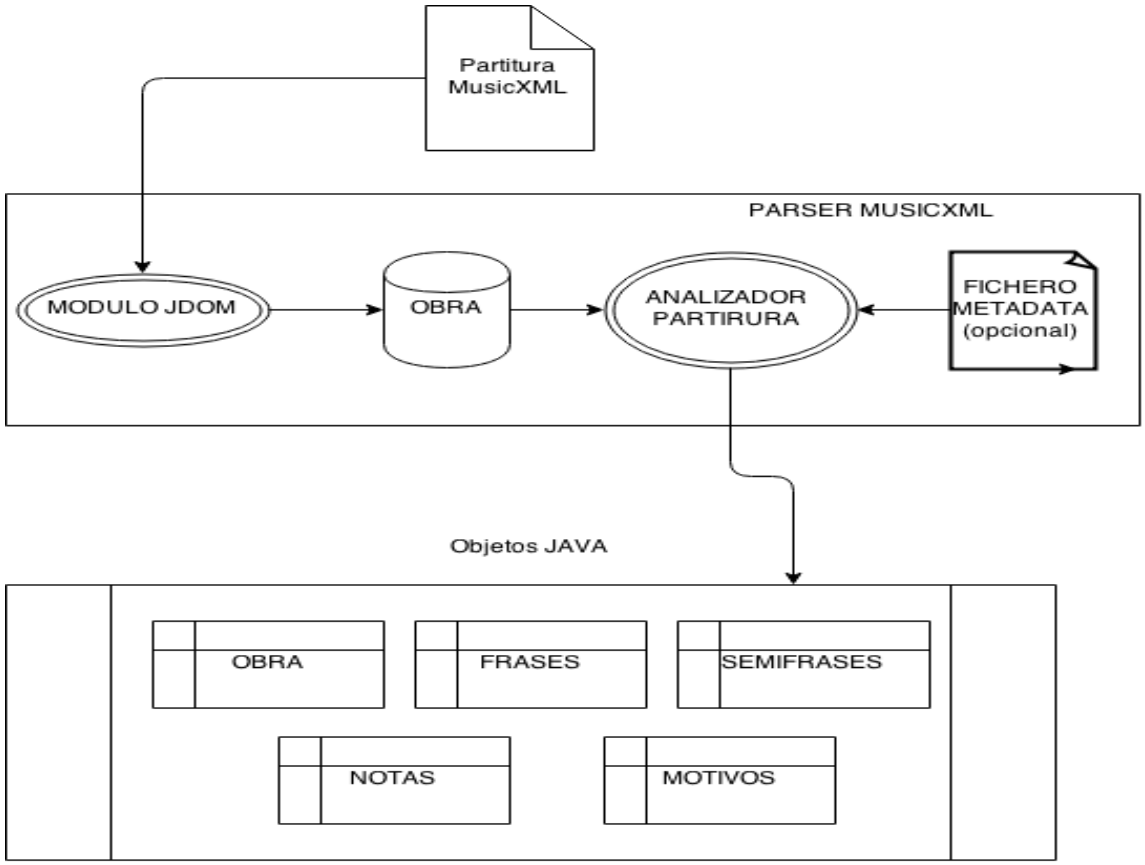

Figura 18: Arquitectura del parser musicXML

4.2.2. Sistema experto JESS

La entrada de este módulo será el conjunto de objetos que contienen la información de la partitura (clave, compás, voces, frases, semifrases y motivos,) . Así, se crea la salida: el fichero de hechos generado en *CLIPS*.
Una vez que los hechos en *CLIPS* son generados, *JESS* carga 3 ficheros:

- Fichero con plantillas: En *JESS*, cada hecho ha de tener una plantilla. Las plantillas son la estructura de cada hecho, siendo el fichero de hechos el que da valor a dicha estructura.
- Fichero de reglas: Contiene el fichero de reglas definidas para el sistema experto y que son expuestas en la sección 4.3.
- Fichero de hechos: Son los hechos que representan una partitura antes de ser aplicada ninguna regla.

A continuación se muestra una figura resumiendo la arquitectura de este módulo:

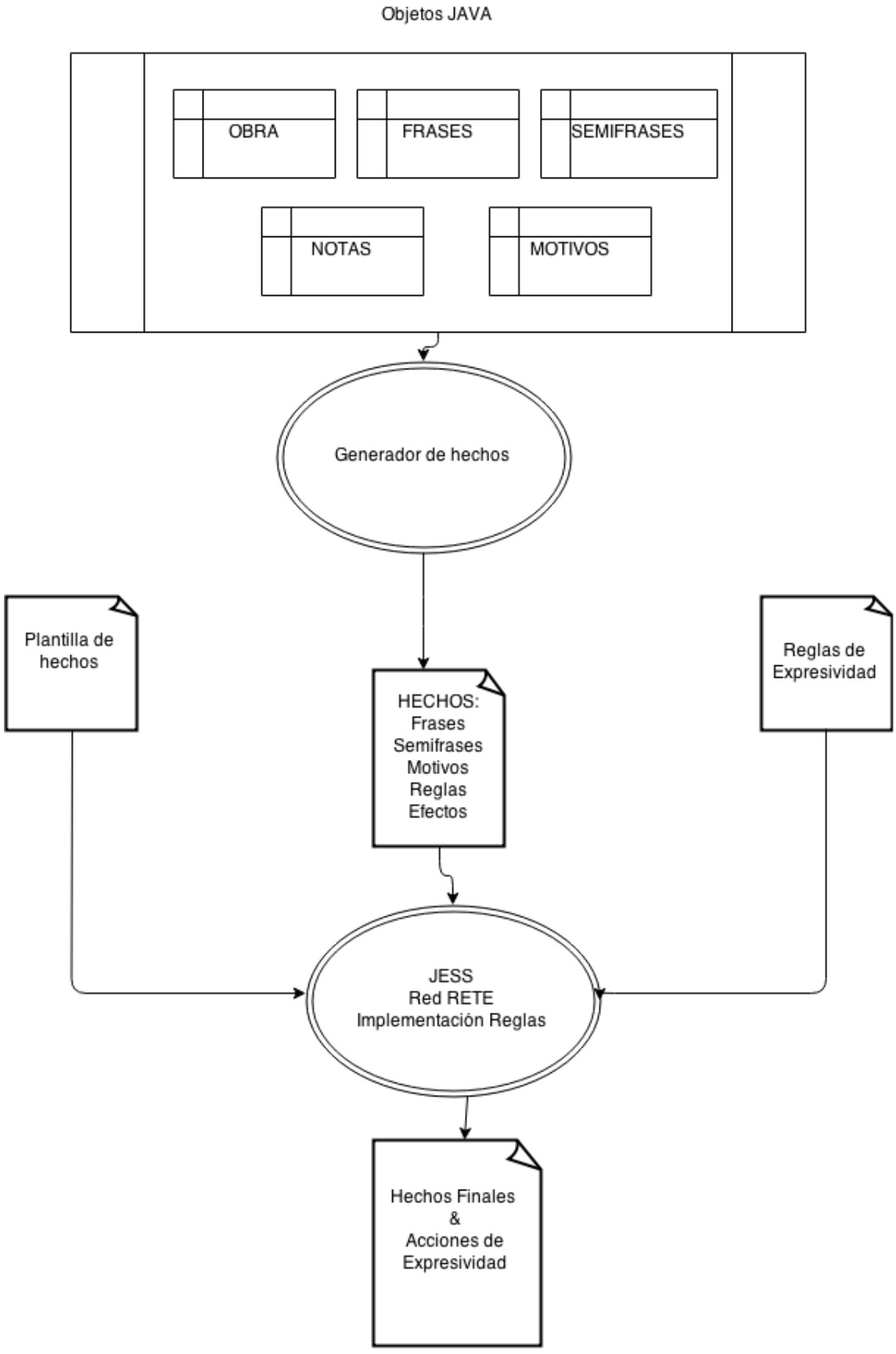

Figura 19: Arquitectura del Sistema Experto

4.2.3. Reproductor MIDI

- Entrada: Hechos modificados por *JESS*.
- Salida: Archivo en formato *MIDI*. En esta última parte, el sistema lee los hechos modificados por *JESS* y crear una lista de notas interpretadas las cuales ya tienen la intensidad (volumen), duración y efectos aplicados. Por tanto, la última parte es hecha por el secuenciador *MIDI*, que se encarga de escribir la melodía acorde a esta lista de notas. El secuenciador usará las siguientes características:
  - Instrumento: violín.
  - Canales: Tantos como voces haya.
  - Tempo: definido por el usuario.
  - Mensajes: NOTE_ON y NOTE_OFF por cada nota a tocar.

A continuación se muestra una figura resumiendo la arquitectura del módulo *MIDI*:

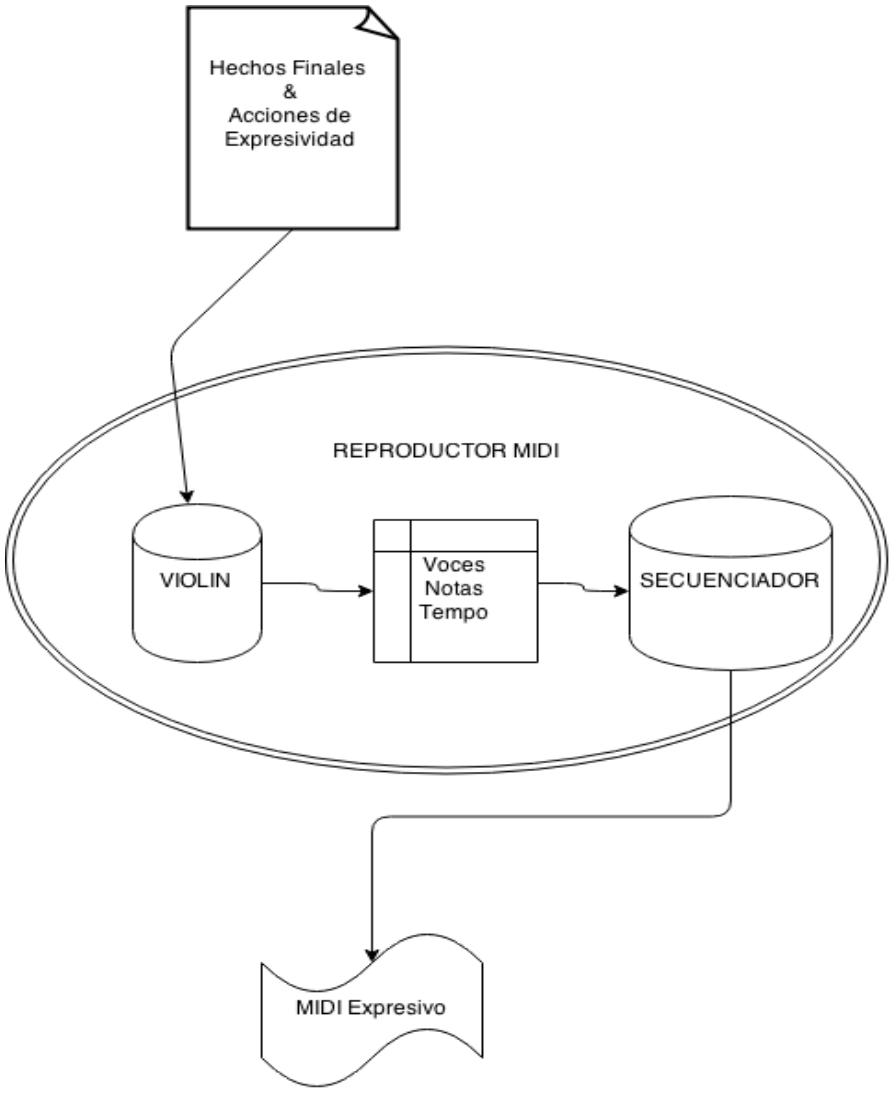

Figura 20: Arquitectura del Reproductor MIDI

4.3. Conocimiento Experto

A continuación se exponen las reglas que serán usadas por el sistema experto y que aportarán la expresividad a la melodía. Así mismo, se muestra su traducción al lenguaje *CLIPS*.

Las reglas han sido creadas a través de una entrevista a un experto con los siguientes datos:

- Nombre y apellidos: Miguel Ramón Gil Menéndez
- DNI: 47468064C
- Título: Título Profesional de Música por el Conservatorio Manuel de Falla de Alcorcón.

Las preguntas realizadas han sido las siguientes:

- ¿Qué técnicas del violín relacionadas con la variación de intensidad definirías como más características del Barroco?
- ¿Qué técnicas del violín relacionadas con la variación del tiempo definirías como mas características del Barroco?

Así, tras la entrevista, la definición de reglas es expuesta en la siguiente tabla:

ID Regla	**Nombre Regla**	**Tipo**	**Descripción**
RI-01	motivo1 ascendente motivo2 descendente	Intensidad	Si la progresión de una semifrase está formada por un motivo ascendente y posteriormente por un motivo descendente, el primero sufrirá un cambio de intensidad de forma que vaya aumentando hasta el final de éste. El segundo comenzará en la intensidad alcanzada y empezará a disminuir hasta el final
RI-02	motivo1 descendente motivo2 ascendente	Intensidad	Si la progresión de una semifrase está formada por un motivo descendente y posteriormente por un motivo ascendente, el primero sufrirá un cambio de intensidad de forma que vaya disminuyendo hasta el final de éste. El segundo comenzará en la intensidad alcanzada por el primero y empezará a aumentar hasta el final.
RI-03	disminuir motivo repetido	Intensidad	Si un motivo se repite (incluyendo sus notas y figuras, no solo su estructura), el primero se ejecutará con mayor intensidad que el segundo)
RI-04	acentuar_notas	Intensidad	Si se repiten 2, 3 o más notas, la intensidad de la primera se aumenta o acentúa (símbolo >), disminuyendo la de las siguientes
RI-05	motivo2 mas altura motivo1	Intensidad	Cuando un motivo tiene una altura mayor al motivo anterior (esto es, las notas son más agudas), el volumen con el que éste se toca ha de ser mayor

RI-06	motivo1 mas altura motivo2	Intensidad	Cuando un motivo tiene una altura menor al motivo anterior (esto es, las notas son más graves), el volumen con el que éste se toca ha de ser menor
RT-01	aplicar ritardando final frase	Tiempo	Si una frase es terminativa, las notas anteriores también han de sufrir un retardo en el tiempo.
RT-02	aplicar ritardando final obra	Tiempo	Cuando la obra termina, se aplica un retardo en el tiempo.

Cuadro 4: Tabla con las reglas de expresividad extraídas

4.4. Diseño del sistema experto

En esta sección se describe el diseño de la base de hechos y la base de reglas del sistema experto para *JESS*.

- Base de hechos:

  1. frase: Representa una frase musical.
  2. semifrase: Representa una semifrase musical.
  3. motivo: Representa un motivo musical. El motivo es la base para la aplicación de expresividad, por tanto, a continuación se detallan los elementos característicos de éste:

     a) altura: Define la altura del motivo. La altura de la nota viene definida de la siguiente forma: Dada la siguiente imagen de un teclado [27]:

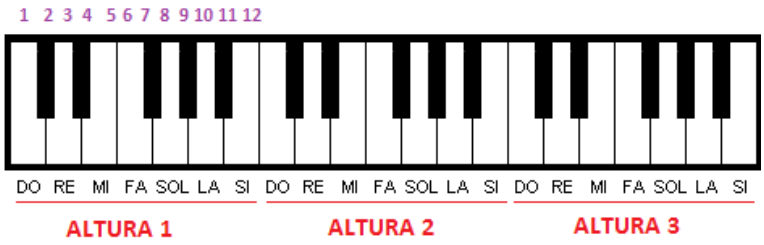

Figura 21: Notas musicales en un teclado.

Se puede observar que las notas (DO-RE-MI-FA-SOL-LA-SI) se repiten cíclicamente en el teclado. Cada ciclo sería una altura y la enumeración de la parte superior de la imagen la posición de la nota de la tecla en cada altura. Recordar que aunque existan 7 nombres para las notas, existen las alteraciones (bemol y sostenido) y es por eso por el que hay un total de 12 sonidos.

b) dirección: Indica la progresión de la altura de las notas que sigue el motivo. Se calcula como la suma de las distancias de las notas. Hay tres tipos:

1) Ascendente: Es la progresión cuya suma de las notas es mayor que cero. Por ejemplo:

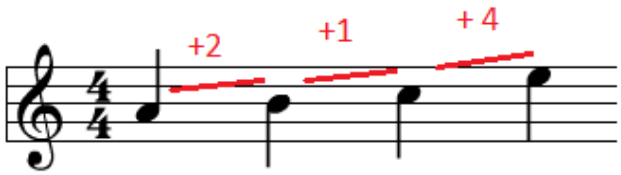

Figura 22: Motivo ascendente

La suma de las distancias sería +7.

2) Descendente: Es la progresión cuya suma de las notas es menor que cero. Por ejemplo:

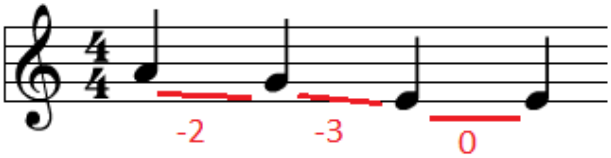

Figura 23: Motivo descendente

La suma de las distancias sería -5.

3) Plano: Es la progresión cuya suma de las notas es igual que cero. Por ejemplo:

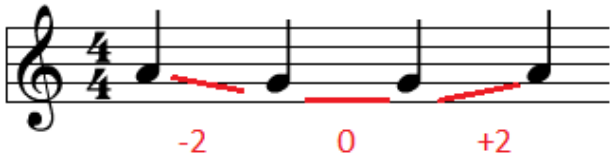

Figura 24: Motivo plano

La suma de las distancias sería 0.

c) base: Define la base de volumen de la que parte el motivo.

d) intensidad: Define, por cada nota del motivo, la intensidad (volumen) a la que es tocada.

e) duración: Define la duración de cada nota del motivo.

f) distancias: Define la distancia en semitonos con la nota siguiente del motivo (ver definición de alteración en la sección 2.1.1). A continuación se muestra gráficamente para un mejor entendimiento:

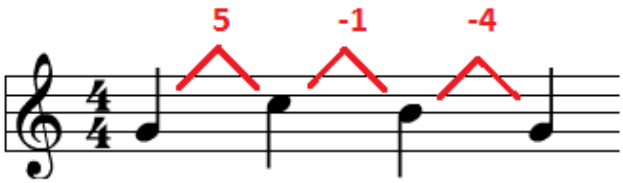

Figura 25: Distancia entre notas

g) estructuraNotas: Consiste en una codificación de las distancias de las notas (punto anterior). Para cada distancia, se aplican las siguientes reglas:

1) Si es > 0 : el elemento del array *estructuraNotas* será 2.
2) Si es < 0 : el elemento del array *estructuraNotas* será 1.
3) Si es $== 0$: el elemento del array *estructuraNotas* será 0.

El propósito de este elemento es el de encontrar posibles motivos cuya estructura sea similar entre notas consecutivas y que puedan mantener una relación ascendente (la segunda nota tiene mayor altura que la primera), descendente (la segunda nota tiene mayor altura que la primera), o plana (las dos notas tienen la misma altura).
Por ejemplo, este motivo:

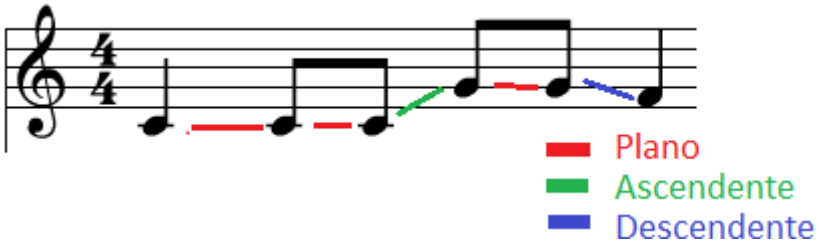

Figura 26: Motivo con estructura de notas 1

Tendría la misma estructura que éste:

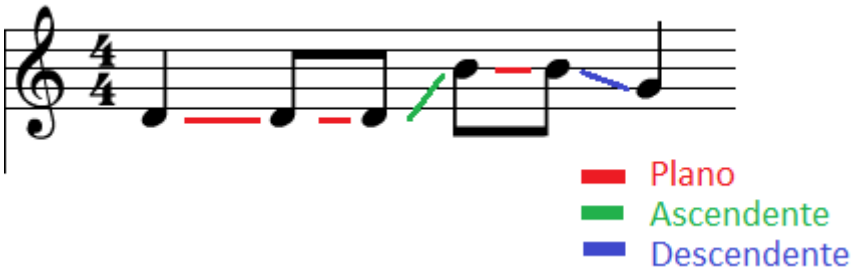

Figura 27: Motivo con estructura de notas 2

Es decir, aunque las notas sean diferentes, es posible apreciar que la relación ascendente/descendente es la misma.

h) estructuraFiguras: Es similar a la propiedad anterior pero éste está relacionado con la duración de las notas. Es decir, *estructuraNotas* tiene que ver con la altura de las notas y *estructuraFiguras* con la duración de éstas.

i) notas: Corresponden al conjunto de notas del motivo. La codificación consiste en concatenar el nombre de la nota + la altura de la nota + la figura de la nota. Las codificaciones del nombre de la nota y la figura están explicadas en el anexo 1 cuadros 18 y 19.

4. reglaAplicada: Representa a una regla ya aplicada y sirve para llevar un seguimiento de reglas que ya han sido aplicadas al motivo evaluado y que permita una jerarquía de reglas (explicada en el siguiente punto). Sus atributos son:

   a) nombre: Contiene el nombre de la regla ya aplicada.
   b) motivo: Contiene el id del motivo sobre el que se ha aplicado la regla.

5. efectoNota: Representa a un efecto que se quiere aplicar sobre una nota de un motivo, como por ejemplo acentuar la nota o ejecutar un *ritardando*. Sus atributos son:
   a) nombre: Contiene el nombre del efecto.
   b) motivo: Contiene el identificador del motivo sobre el que se quiere realizar el efecto.
   c) nota: Contiene el identificador de la nota sobre el que se quiere realizar el efecto.

- Base de reglas: A continuación se detalla la codificación de las reglas extraídas a partir del conocimiento experto, explicando la acción - efecto de cada una de ellas. Las reglas son listadas por los identificadores definidos en la tabla 4

  - RI-01: Esta regla modifica la intensidad de dos motivos consecutivos teniendo en cuenta la dirección de éstos, el primero es aumentado y el segundo disminuido.
    - Acción:
      - Si el identificador del motivo 1 es 1 unidad menor al identificador del motivo 2.
      - Si la dirección del primer motivo es ascendente y la dirección del segundo motivo es descendente.
    - Efecto:
      - Modificar intensidad al motivo 1, subiendo el volumen progresivamente desde la primera nota hasta la última nota del motivo.
      - Modificar intensidad al motivo 2, bajando el volumen progresivamente desde la primera nota hasta la última nota del motivo.
  - RI-02: Esta regla modifica la intensidad de dos motivos consecutivos teniendo en cuenta la dirección de éstos, el primero es disminuido y el segundo es aumentado.
    - Acción:
      - Si el identificador del motivo 1 es 1 unidad menor al identificador del motivo 2.
      - Si la dirección del primer motivo es descendente y la dirección del segundo motivo es ascendente.
    - Efecto:
      - Modificar intensidad al motivo 1, bajando el volumen progresivamente desde la primera nota hasta la última nota del motivo.
      - Modificar intensidad al motivo 2, subiendo el volumen progresivamente desde la primera nota hasta la última nota del motivo.
  - RI-03: Esta regla modifica la intensidad de dos motivos consecutivos teniendo en cuenta la estructura de éstos, el primero es aumentado y el segundo es disminuido.
    - Acción:
      - Si el identificador del motivo 1 es 1 unidad menor al identificador del motivo 2.

- ◇ Si la estructura de ambos motivos es igual.
- Efecto:
  - ◇ Modificar intensidad al motivo 1, subiendo el volumen de la base del motivo.
  - ◇ Modificar intensidad al motivo 2, bajando el volumen de la base del motivo.

- RI-04: Esta regla modifica la intensidad de un motivo, teniendo en cuenta las notas que contiene. Si se repiten 2, 3 o más notas, la intensidad de la primera se aumenta o acentúa (símbolo >), disminuyendo la de las siguientes. La figura 28 hace referencia a un compás sin expresividad. La figura 29 es el mismo fragmento pero con la expresividad aplicada al haber sido analizado.

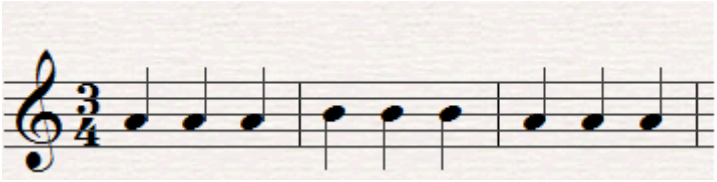

Figura 28: Fragmento a ser acentuado

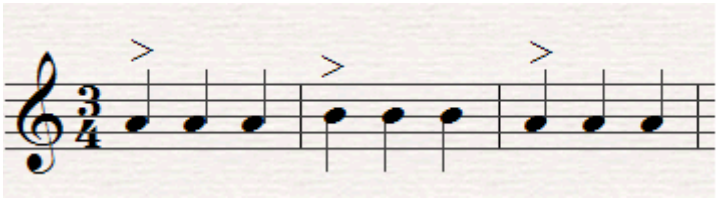

Figura 29: Fragmento anterior interpretado

- Acción:
  - ◇ Si el motivo tiene todas las notas con la misma figura de tiempo.
  - ◇ Si el motivo tiene todas las notas de la misma altura.
- Efecto:
  - ◇ Acentuar la primera nota de cada estructua encontrada en el motivo (4 semicorcheas, 3 negras, etc).

- RI-05: Esta regla modifica la intensidad de dos motivos, teniendo en cuenta la altura de cada uno de ellos.
  - Acción:
    - ◇ Si el identificador del motivo 1 es 1 unidad menor al identificador del motivo 2.
    - ◇ Si la altura del motivo 2 es mayor a la del motivo 1.
  - Efecto:
    - ◇ Aumentar la intensidad del segundo motivo, aumentando el volumen al doble de la base de éste.

- RI-06: Esta regla modifica la intensidad de dos motivos, teniendo en cuenta la altura de cada uno de ellos.
  - Acción:
    - Si el identificador del motivo 1 es 1 unidad menor al identificador del motivo 2.
    - Si la altura del motivo 1 es mayor a la del motivo 2.
  - Efecto:
    - Aumentar la intensidad del primer motivo, aumentando el volumen al doble de la base de éste.
- RT-01: Esta regla el tiempo de un motivo, teniendo en cuenta si éste es el que termina una frase.
  - Acción:
    - Si el el motivo es el último de una frase.
  - Efecto:
    - Aplicar el efecto *ritardando*
- RT-02: Esta regla el tiempo de un motivo, teniendo en cuenta si éste es el que termina una obra.
  - Acción:
    - Si el el motivo es el último de la obra.
  - Efecto:
    - Aplicar el efecto *ritardando*

Cabe destacar que estas dos últimas reglas inicialmente se codificaron por separado, pero posteriormente se observó que ambas reglas pueden ser representadas en la misma, ya que la terminación de una frase está contenida en la terminación de la obra.

4.5. Implementación

En esta sección se detallan los paquetes y clases del sistema a través de los cuales la información fluye haciendo hincapié en los algoritmos usados para representar las estructuras musicales y que serán usadas por el sistema experto. El diagrama de flujo mostrando la relación de los paquetes es el siguiente:

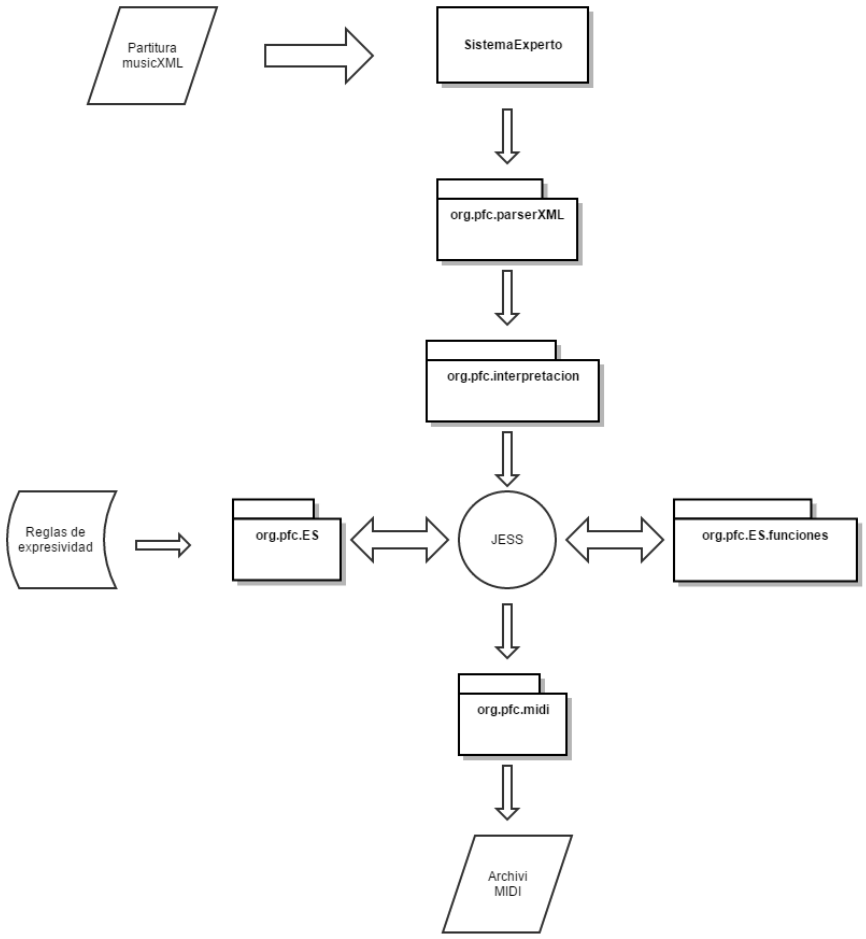

Figura 30: Diagrama UML del paquete org.pfc.parserXML

4.5.1. Parser de partituras XML

El parser de partituras en formato *musicXML* está programado en el paquete *org.pfc.parserXML*. El diagrama de flujo que conecta los distintos paquetes es el siguiente:

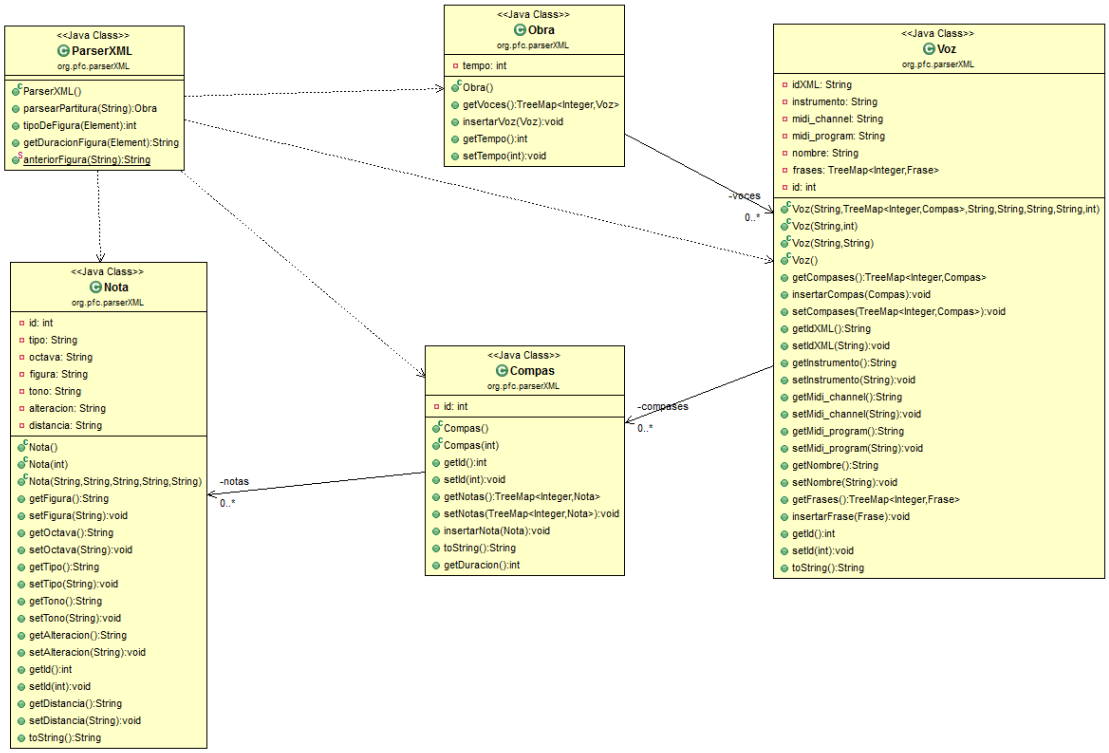

Figura 31: Diagrama UML del paquete org.pfc.parserXML

Como se puede observar, la clase indivisible es la clase *NOTA*. La cual posee los siguientes atributos:

- id: Es el identificador de la nota. Es único e incremental, es decir, la primera nota de la melodía será el id 0, la siguiente el id 1, y así sucesivamente.
- tipo: Indica si es un tono (sonido), silencio, o acorde (varios sonidos simultáneos).
- octava: Indica la *octava* a la que pertenece la nota. La *octava* fue explicada en la sección 2.1.1.
- figura: Indica la figura de la nota: negra, blanca, redonda, etc. La representación está expuesta en el anexo 1, cuadro 18.
- tono: Indica la representación en formato *String* de la nota. La representación es el resultado de concatenar los siguientes elementos:
  - Nombre de la nota en notación inglesa (ver anexo 1, cuadro 19).
  - Octava a la que pertenece.
  - Figura de la nota.
- Alteración: Su valor es 0 si no existe alteración, -1 si la alteración es un bemol o 1 si la alteración es un sostenido.
- Distancia: Muestra la distancia entre esta nota y la siguiente. La distancia es el número de semitonos que hay entre ambas notas.

La clase Compás contiene una lista de las notas que contiene, así como el identificador del compás (entero incremental).
La clase Voz contiene elementos extraídos de la partitura en formato *musicXML*, que luego serán usados por el reproductor *MIDI*, estos son:

- midi_channel: Indica el canal por el cual se escribirá la melodía.
- midi_program: Mapea el instrumento (en este caso violín) con el identificador *MIDI* del instrumento (en este caso 47)

Los *atributos idXML, instrumento y nombre* son de caracter informativo para pruebas. También contiene el conjunto de frases de la obra. Las frases son representadas en el paquete *org.pfc.interpretacion*. El diagrama UML de éste es el siguiente:

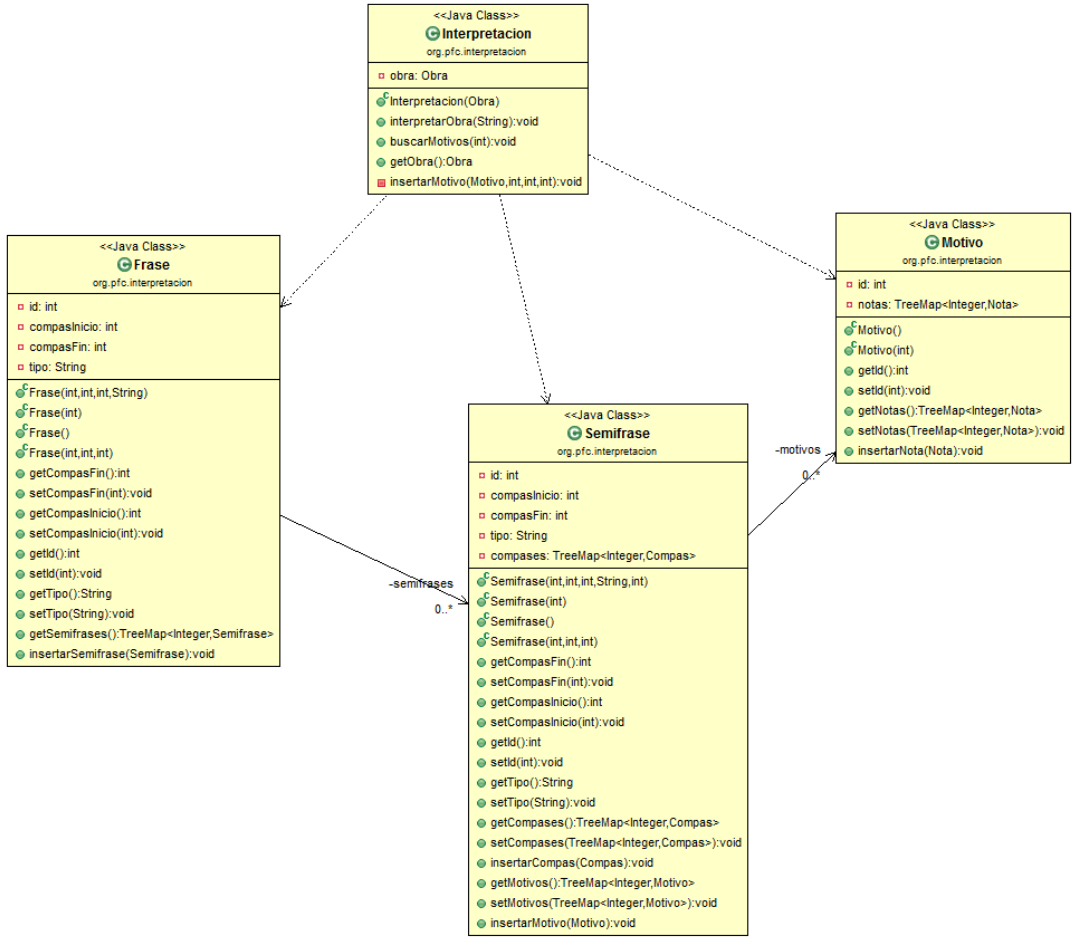

Figura 32: Diagrama UML del paquete org.pfc.interpretacion

La clase *Interpretación* es el punto de entrada para la intepretación de la partitura. Así, el constructor de ésta es una instancia de la clase *Obra* que contiene los objetos resultado del parseo de la partitura *musicXML*. Mediante el diagrama, es posible observar las siguientes características:

- La clase *Motivo* es la unidad básica de la intepretación, conteniendo así las notas (similar a la clase *Compás*).

- La clase *Semifrase* contiene de 0 a N motivos.
- La clase *Frase* contiene de 0 a N semifrases.
- Los límites de las frases y semifrases están marcadas por los atributos *compasInicio* y *compasFin* de las clases *Frase* y *Semifrase* respectivamente.
- El atributo tipo contendría el tipo de frase o semifrase encontrado (terminativa, afirmativa, negativa, cadencial, etc) y sería usado para una interpretación más avanzada que no entra en el objetivo de este proyecto (ver sección 6.1).

4.5.2. Sistema experto JESS

El sistema experto *JESS* está implementado en los paquetes *org.pfc.ES* y *org.pfc.ES.funciones*. El diagrama UML que muestra ambos paquetes es el siguiente:

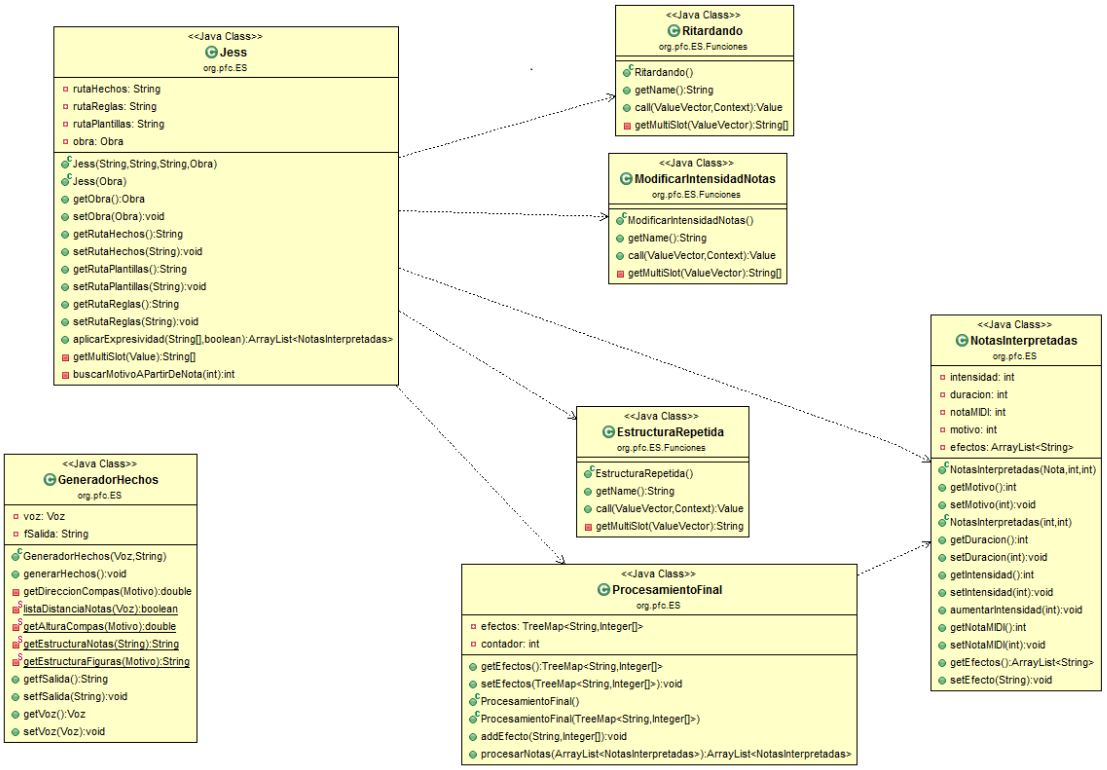

Figura 33: Diagrama UML del paquete org.pfc.ES

El sistema experto está implementado en dos clases fundamentales:

- *GeneradorHechos.java*: Como se puede observar en el diagrama, al constructor de esta clase se le pasa una instancia de la clase *Voz*. Esta clase es la encargada por tanto de codificar cada una de las voces de la partitura en formato *CLIPS*. En la sección 4.4 se muestran los hechos generados. Aquí se explican las siguientes partes de la voz que son representados:
  - frase: Este hecho representa una frase de la partitura, se escribe su id (identificador) y el tipo de la frase.

- semifrase: Este hecho representa una semifrase de la partitura, se escribe su id (identificador), el tipo de la semifrase y la frase a la que pertenece (escribiendo el id de ésta).
- motivo: Este hecho representa un motivo de la partitura. En esta parte es donde se implementa la mayor parte de la lógica de la expresividad ya que la mayoría de las reglas de expresividad usan el motivo como elemento a analizar. A continuación se explican los atributos más relevantes para la expresividad y su cálculo:
  1. id: identificador numérico del motivo.
  2. frase: frase a la que el motivo pertenece.
  3. semifrase: semifrase a la que el motivo pertenece.
  4. altura: Representa la altura del compás. Se calcula como la media de las alturas de cada una de sus notas. La altura de la nota se calcula de la siguiente manera: $12 * y + x$ donde y es la altura de la nota y x la posición de la nota (1-12).
- distancias: Es una array de enteros en el que se representa la diferencia de altura entre dos notas consecutivas.
- dirección: Indica la progresión de la altura de las notas que sigue el motivo. Se calcula como la suma de las distancias de las notas. La dirección ascendente corresponde a un valor final mayor que 0, descendente menor que 0 y plano igual a 0.
- base: Es la base de volumen del motivo. Es decir, sin expresividad, el motivo se tocaría con la intensidad especificada. El generador de hechos establece el valor 70 (valor para el protocolo *MIDI*) y sirve para que posteriormente las reglas lo modifiquen si aplica.
- intensidad: Es un array de números que representa la intensidad con la que se toca cada nota del motivo. El generador de hechos establece un valor de 0 por cada nota para que posteriormente las reglas de expresividad la modifiquen cuando aplique. Así, el volumen final de la nota en la posición i del motivo será: $b + x[i]$ donde b es la base del compás y x el array de intensidades.
- duracion: Es un array de números que representa la duración de cada nota del motivo. El generador de hechos establece el valor 1.0 por cada nota para que posteriormente las reglas de expresividad lo modifiquen cuando aplique. Así, la duración final de la nota en la posición i del motivo será: $f * x[i]$ donde f es la duración de la figura de la nota (negra, corchea, blanca, etc) y x el array de duraciones.

- *JESS.java*: Tras la generación de hechos, el siguiente paso es aplicar la expresividad (método *Jess.aplicarExpresividad()*) de la siguiente forma:

  1. Se cargan los 3 ficheros necesarios para crear la red *Rete*: las plantillas de los hechos, las reglas de expresividad y los hechos generados. Los dos primeros serían siempre fijos y el último es el que dependería de la partitura leída.
  2. Una vez cargados, se añaden a la red *Rete* las funciones de usuario. Estas funciones lo que hacen es ser ejecutadas cuando alguna regla de expresividad tenga que ser aplicada. Es decir, si la parte *if* de la regla se cumple, estas funciones serían

llamadas en la parte *then* de la regla. A continuación se exponen las reglas de usuario:

- *ModificarIntensidadNotas.java*: Se encarga de modificar la intensidad de las notas y motivos a partir de los argumentos de las reglas de expresividad. Los argumentos son:
  a) Dirección del motivo (-1, 0 ó 1)
  b) Si el motivo a cambiar es el primero o el segundo (para reglas de expresividad ascendentes o descendentes, ver sección 4.3).
  c) Valor de la base del motivo a cambiar.
  d) Valor de la base del otro motivo.
  e) Intensidades de las notas del motivo a cambiar
  f) Intensidades de las notas del otro motivo a cambiar
  g) Valor de la base del tercer motivo.
  De esta forma, todas las reglas de expresividad definidas con relación a la modificación de la intensidad de la obra usarán esta función para hacerlo.
- *EstructuraRepetida.java* : Esta función es llamada por las reglas *JESS* para comprobar si, al inspeccionar un motivo, existe otro con la misma estructura (ver el punto motivo:estructuraNotas de la sección 4.4).
- *Ritardando.java*: Esta función aplica el *ritardando* a un motivo cuando las reglas de expresividad así lo indican.

3. Tras la adicción de reglas, el siguiente paso es ejecutar la red *Rete*, aplicando así las reglas de expresividad.
4. Una vez que el proceso ha acabado, se recorren todos los motivos de la obra, omitiendo las frases y semifrases ya pues esos hechos solo eran de interés para la aplicación de expresividad y se crea por cada nota una instancia de la clase *NotaInterpretadas*.

- *NotaInterpretadas.java*: Esta clase representa a cada nota de la melodía ya interpretada contiene todo lo necesario para que el reproductor MIDI escriba la melodía: intensidad, duración, la nota del protocolo *MIDI* y los efectos que posee. En el proyecto se ha desarrollado solo un efecto: el de acentuar las notas. Los efectos son introducidos por la parte *then* de la regla "acentuar_notas". Los efectos son aplicados en la clase *ProcesamientoFinal*.

- *ProcesamientoFinal.java*: Esta clase aplica los efectos encontrados en cada una de las instancias de la clase *NotaInterpretada*. En esta caso, el efecto "acento" se realiza de la siguiente forma: si la nota interpretada lo posee, la intensidad de está se ve incrementada en 40 unidades (valor para la salida *MIDI*).

Así, se ha descrito el proceso que sigue el sistema experto desde que recibe las notas de la partitura hasta que las modifica.

4.5.3. Reproductor MIDI

El reproductor *MIDI* es implementado en el paquete *org.pfc.midi* y su diagrama UML es el siguiente:

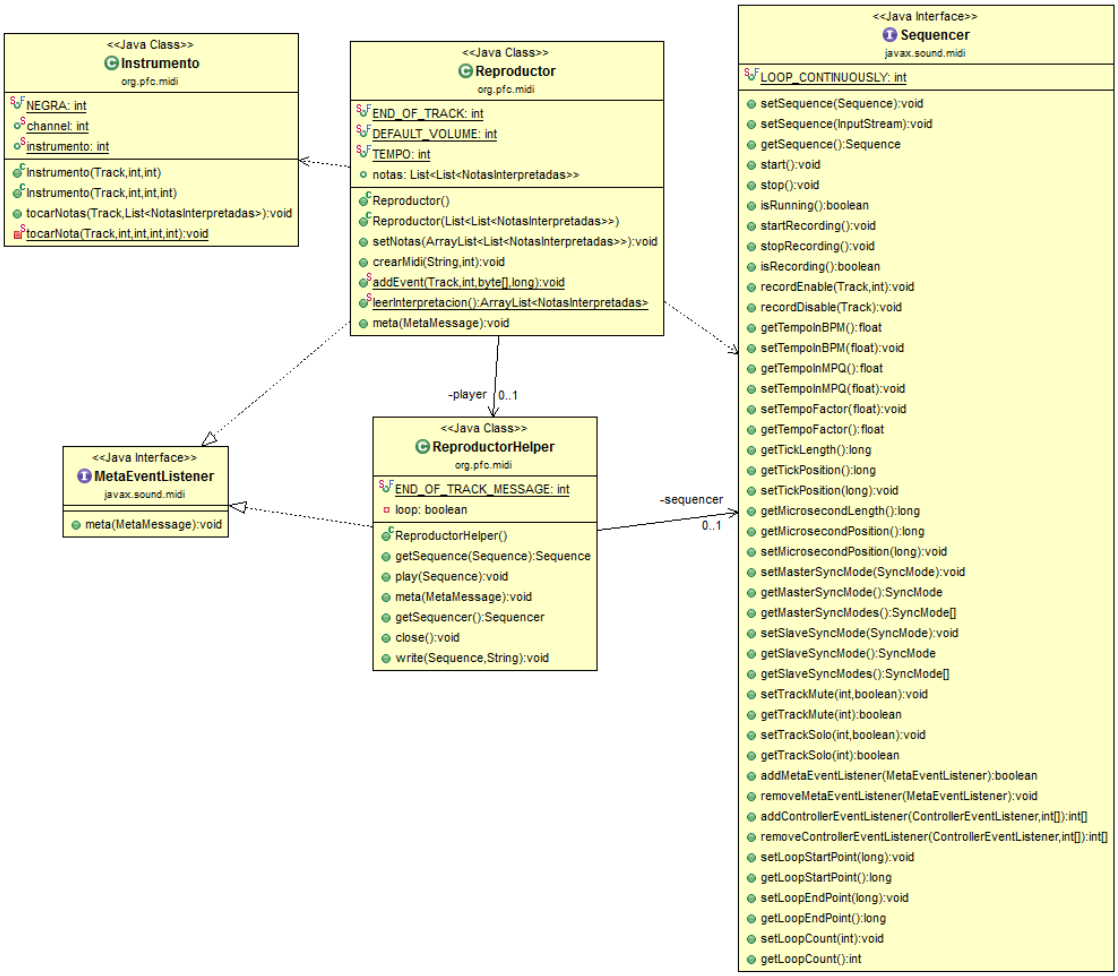

Figura 34: Diagrama UML del paquete org.pfc.midi

La clase principal es *Reproductor*. Esta clase es la encargada de crear el fichero *MIDI* a partir de las notas interpretadas. Todo se concentra en el método *crearMidi()*. A continuación se detallan los pasos:

- A partir de los argumentos de entrada (fichero de salida y el *tempo* de la obra), se crea una secuencia, instancia de la clase *Sequence*.
- Después, por cada voz:
  - Se crea una instancia de la clase *Track*
  - Se inserta el mensaje *MIDI* que indica el comienzo de los mensajes. Definiendo el *tempo* de la obra.
  - Se inserta el mensaje *MIDI* que indica que el instrumento que ha de sonar es el violín.
  - Después, es el instrumento, instancia de la clase *Instrumento*, quién escribe los mensajes tipo *NOTE_ON* y *NOTE_OFF* de *MIDI* para representar cada una de las notas. Estos mensajes van encapsulados en instancias de la clase *MidiEvent*.

- Una vez escritas las notas en el fichero *MIDI*, es una instancia de la clase *Reproductor-Helper* quién o bien escribe el fichero *MIDI* creado en memoria a disco o lo reproduce directamente, dependiendo de la opción elegida por el usuario en la ejecución del programa (ver Anexo 2, Manual de Usuario).

5. Experimentación y resultados

En esta sección se exponen los distintos experimentos realizados y los resultados obtenidos. Para ello, se han elegido distintas obras para probar las siguientes características musicales:

- Modificación de la intensidad de la obra: Incluye los cambios de volumen.
- Modificación de la variación del tiempo de la obra: Incluye cambios en la duración de los motivos.
- Modificación de la intensidad en obras con más de un violín: Obras con más de un violín serán analizadas para observar la compatibilidad de las variaciones de la intensidad en melodías distintas pero que juntas conforman una obra completa.

Con este fin, los pasos diseñados para la experimentación y observación de los resultados ha sido:

- Selección de una obra concreta: Se elige una obra ya existente de algún compositor o una propia creada y que sirva de entrada del programa. Los dos primeros puntos de la experimentación se analizarán en todas las obras. El tercer y último punto se hará exclusivamente en la obra 4. A modo de resumen, se muestra una tabla 5 con lo mencionado anteriormente:

Experimento	**Comprueba Intensidad**	**Comprueba Duración**	**Comprueba intensidad en más de una voz**
1	Si	Si	No
2	Si	Si	No
3	Si	Si	No
4	Si	Si	Si

Cuadro 5: Tabla resumen con las comprobaciones hechas en cada experimento

- Ejecución del programa: Con los argumentos del programa específicos para la obra elegida, se ejecuta el programa:
- Extracción de logs: El programa imprimirá unos logs que servirán para el posterior análisis en la modificación de la obra resultado de la aplicación de todos y cada uno de los elementos a analizar en esta sección (y que conformarán la salida del programa).
- Comparación entre la salida del programa y logs: Una vez obtenidas ambas fuentes, se comprueba que las modificaciones hechas por el sistema experto sean reconocibles en la melodía de salida. La comprobación ha de hacerse escuchando la melodía, reconociendo la regla aplicada descrita por los logs.

5.1. Obra 1 - Propia

5.1.1. Objetivos

Para este experimento se ha decidido crear una obra propia, intentando representar patrones sobre los que el sistema experto debería actuar de una manera determinada. Es decir, esta

obra ha sido creada para poner a prueba si los patrones musicales programados en el sistema experto son reconocidos. Este experimento se realiza con el objetivo de crear una base sólida de partida para que, en obras más complejas de determinados compositores se apliquen estas reglas. Por tanto, en este experimento se determinará si la salida del programa se corresponde con lo esperado al haber compuesto la obra.

5.1.2. Preparacion

A continuación, se presenta la obra a analizar:

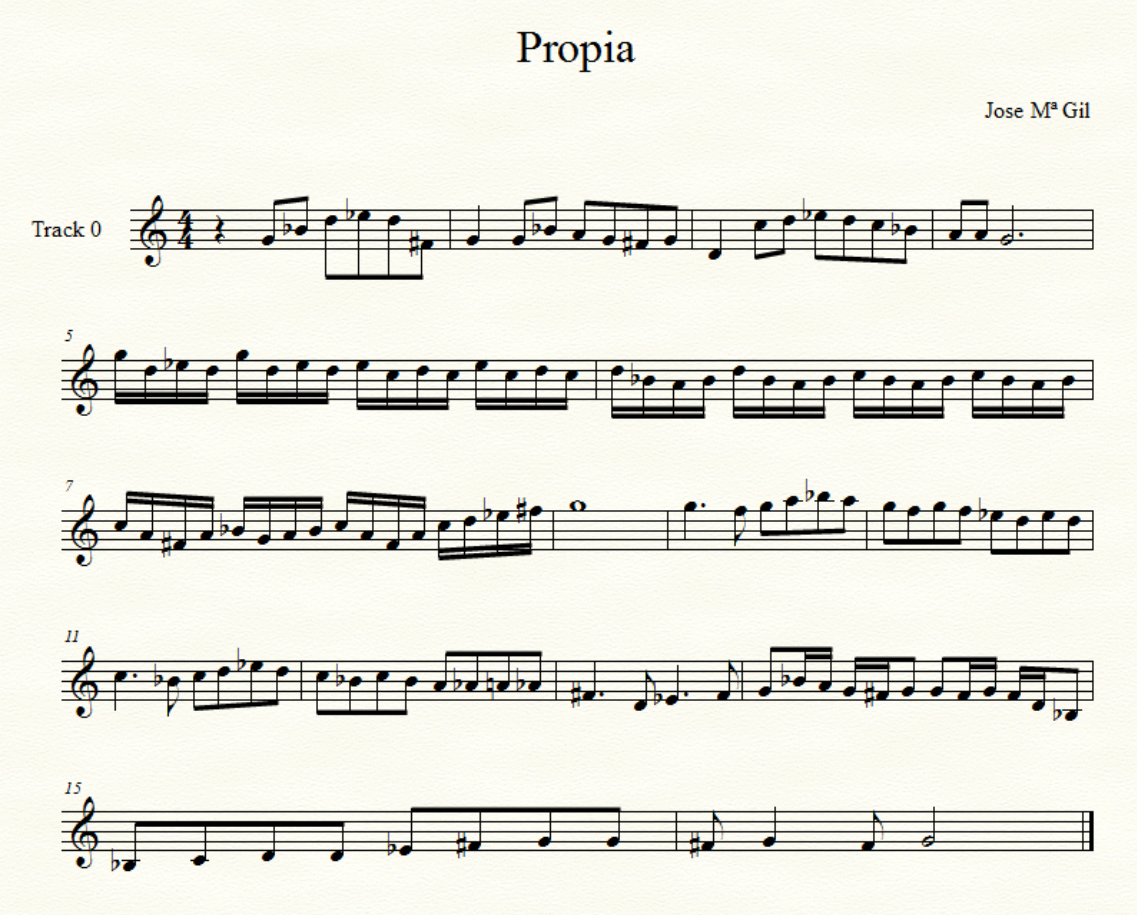

Figura 35: Obra 1 - Partitura

En la obra se aprecian dos frases de 8 compases cada una. Siendo éstas como sigue:
Frase 1

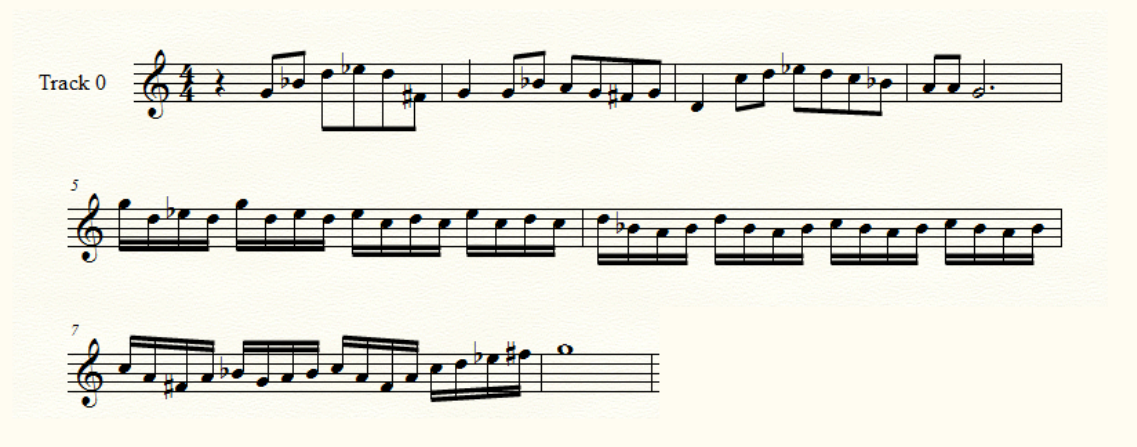

Figura 36: Obra 1 - Frase 1

Frase 2

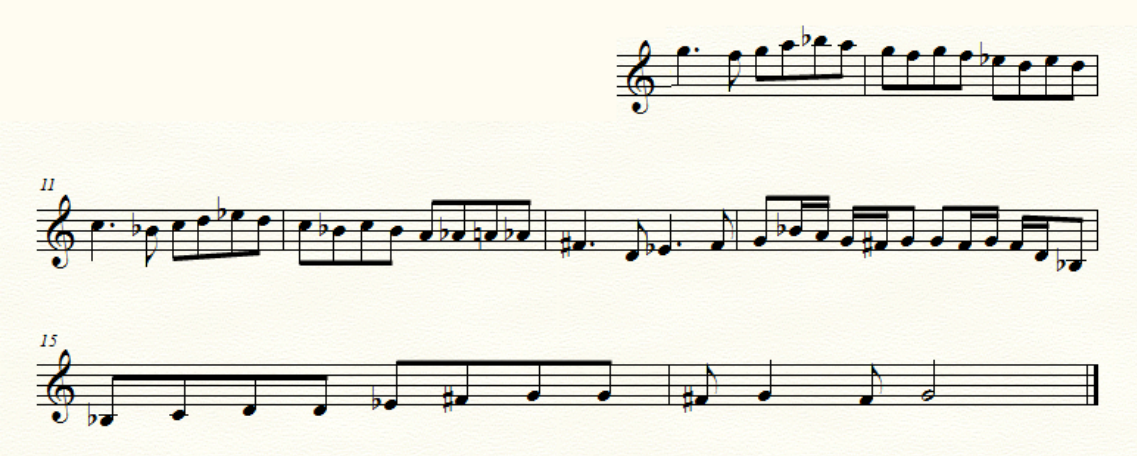

Figura 37: Obra 1 - Frase 2

Tanto la frase 1 como la frase 2 se descomponen en 2 semifrases cada una. Los motivos corresponden a cada compás de la obra.

5.1.3. Resultados y conclusiones

A continuación se muestra una tabla resumen indicando qué regla se ha aplicado a qué motivos:

	RI-01	RI-02	RI-03	RI-04	RI-05	RT-01	RT-02
Motivo 0				X	X		
Motivo 1							
Motivo 2					X		
Motivo 3							
Motivo 4				X	X		
Motivo 5				X	X		
Motivo 6				X		X	
Motivo 7					X		
Motivo 8				X	X		
Motivo 9				X	X		
Motivo 10				X	X		
Motivo 11				X			
Motivo 12	X						
Motivo 13		X			X		
Motivo 14		X		X			
Motivo 15							X

Cuadro 6: Tabla asignación regla-motivo del experimento 1

Conclusiones

La partitura resultante, aplicando dichas reglas, sería:

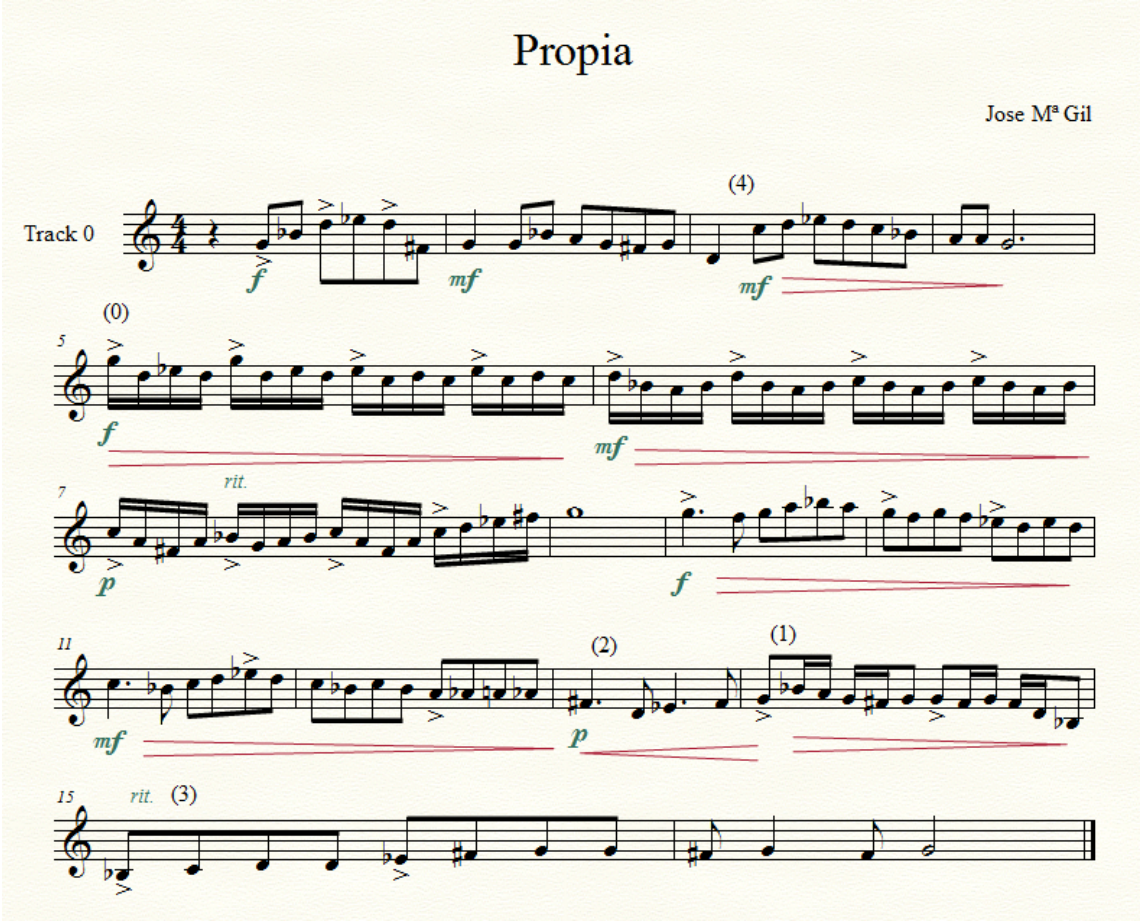

Figura 38: Obra 1 - Partitura con expresividad aplicada

En la partitura, se han representado una regla de cada tipo. A continuación se presenta la relación el nombre de la regla con su número representado en la partitura:

- Se ha aplicado la regla RI-04 sobre el motivo: 4 → (0)
- Se ha aplicado la regla RI-02 sobre el motivo: 13 → (1)
- Se ha aplicado la regla RI-01 sobre el motivo: 12 → (2)
- Se ha aplicado la regla RT-02 sobre el motivo: 15 → (3)
- Se ha aplicado la regla RI-05 sobre el motivo: 2 → (4)

Durante la melodía, se escucha cómo al principio ésta empieza con volumen *forte* para ir disminuyendo la intensidad pues se van encontrando motivos con una altura menor al anterior. En el compás 5 empieza una semifrase donde todas las notas son estructuras de 4 semicorcheas, por lo que se aplica la regla que acentúa las notas (1 de cada 4). A su vez, durante estos compases los motivos van teniendo de nuevo menor altura (por lo que la intensidad disminuye de *forte* hasta *piano*. Al final de la primera frase se aprecia un ritardando que indica el final de ésta. La segunda frase comienza con una estructura completamente distinta, empezando con una intensidad *forte* de nuevo y acentuando las corcheas a lo largo de sus motivos. La segunda semifrase de esta frase comienza piano dado que es la primer semifrase va disminuyendo. Al final de la segunda frase, se aprecia el ritardando que indica el final de la obra.

Por último, que no aparezca la regla *disminuir_motivo_repetido* es normal ya que se puede apreciar que en esta obra ninguna estructura se repite.
Es posible escuchar los resultados en los siguientes ficheros:

- Obra con expresividad: *Resultados/Propia - Con Expresividad.mid*
- Obra sin expresividad: *Resultados/Propia - Sin Expresividad.mid*

5.2. Obra 2 - Mozart

5.2.1. Objetivos

Una estructura de una obra sencilla estaría compuesta por 2 frases de 8 compases cada una. Cada frase contendría 2 semifrases de 4 compases cada una donde cada uno de éstos es un motivo. Dada una obra de esta estructura (Tercer Landler del KV 606 de W.A. Mozart) [36] , se quiere comprobar la aplicación de todas las reglas de expresividad definidas para poder evaluar su impacto en la obra y entender cual es la compatibilidad de las distintas reglas entre sí.

5.2.2. Preparacion

A continuación se muestra la obra a analizar:

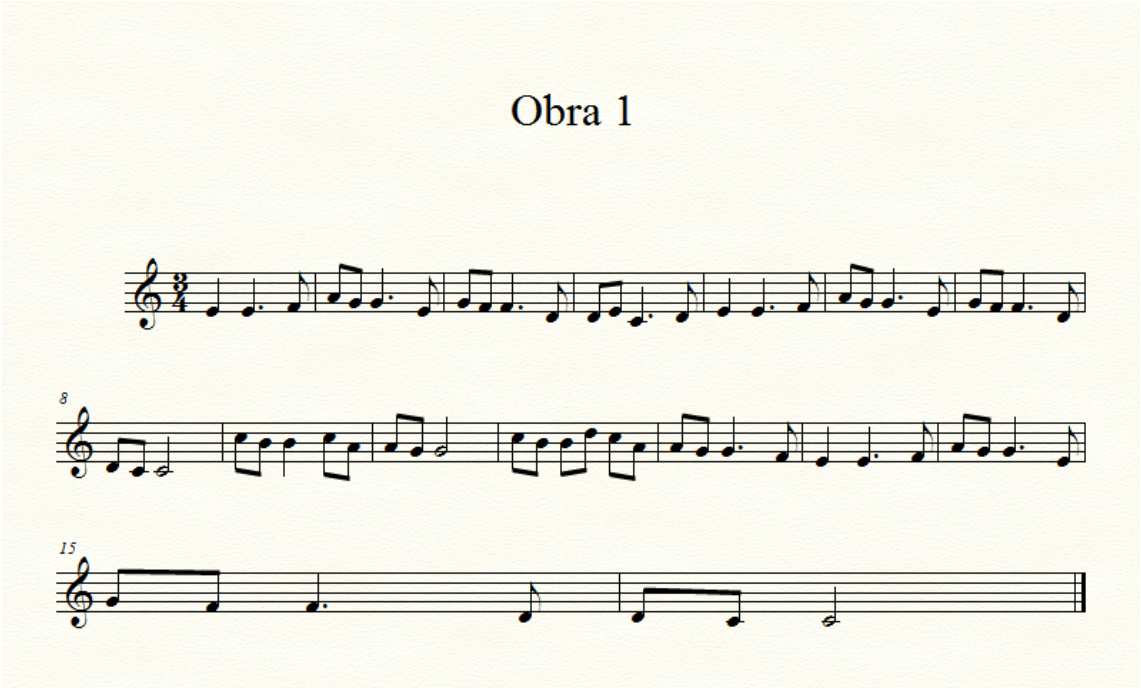

Figura 39: Obra 2

Esta obra servirá como entrada del programa, se observa como cumple con la estructura descrita en el apartado anterior:
Frase 1

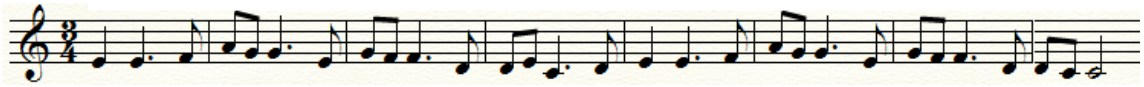

Figura 40: Obra 2 - Frase 1

Frase 2

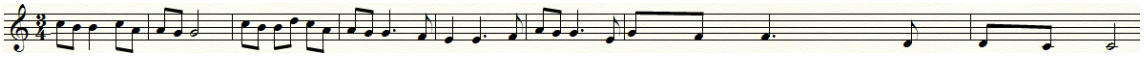

Figura 41: Obra 2 - Frase 2

Tanto la frase 1 como la frase 2 se descomponen en 2 semifrases cada una. Los motivos corresponden a cada compás de la obra.

5.2.3. Resultados y conclusiones

A continuación se muestra una tabla resumen indicando qué regla se ha aplicado a qué motivos:

	RI-01	RI-02	RI-03	RI-04	RI-05	RT-01	RT-02
Motivo 0							
Motivo 1					X		
Motivo 2		X			X		
Motivo 3		X					
Motivo 4	X		X				
Motivo 5	X		X		X		
Motivo 6			X		X	X	
Motivo 7							
Motivo 8					X		
Motivo 9							
Motivo 10					X		
Motivo 11							
Motivo 12	X						
Motivo 13	X				X		
Motivo 14					X		
Motivo 15							X

Cuadro 7: Tabla asignación regla-motivo del experimento 2

Conclusiones

Se puede ver cómo se han aplicado 20 reglas de expresividad, conformando la siguiente partitura conteniendo la notación musical que define los cambios de ritmo e intensidad:

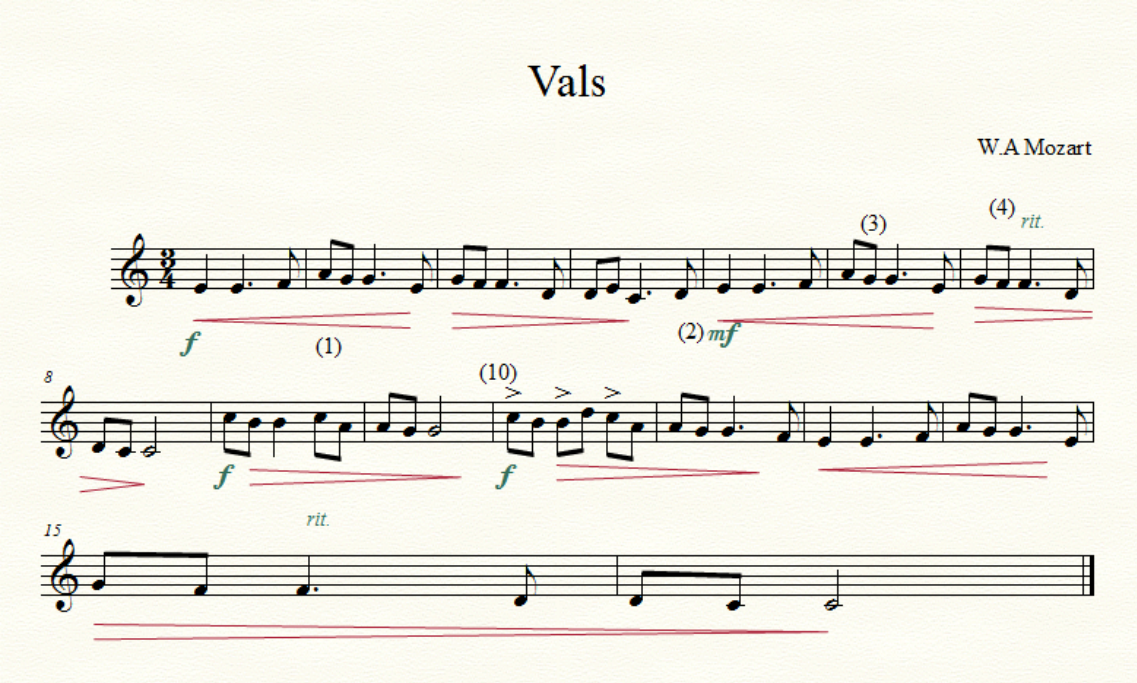

Figura 42: Obra 2 con la expresividad aplicada

En la partitura, se han representado una regla de cada tipo. A continuación se presenta la relación el nombre de la regla con su número representado en la partitura:

- Se ha aplicado la regla RI-05 sobre el motivo: 1 → (1)
- Se ha aplicado la regla RI-03 sobre el motivo: 4 → (2)
- Se ha aplicado la regla RI-01 sobre el motivo: 5 → (3)
- Se ha aplicado la regla RT-02 sobre el motivo: 6 → (4)
- Se ha aplicado la regla RI-04 sobre el motivo: 10 → (5)

La obra comienza con una intensidad normal (*forte*) en la que hay un *crescendo* en el comiendo de la primera semifrase y un *diminuendo* al final. En la segunda semifrase se observa lo mismo, pues la estructura de los motivos es similar. Al final de la primera frase, se observa un pequeño *ritardando* que da paso a la segunda frase. En ésta, se comienza con una intensidad mayor que en el comienzo de la primera frase. En la primera semifrase es observa cómo los dos motivos empiezan con una intensidad *forte* para aplicar un *diminuendo* que da comienzo a la última semifrase, que termina con un *crescendo* más ligero y un *ritardando* más acentuado que indica el final del fragmento.

Es posible escuchar los resultados en los siguientes ficheros:

- Obra con expresividad: *Resultados/Mozart - Con Expresividad.mid*
- Obra sin expresividad: *Resultados/Mozart - Sin Expresividad.mid*

5.3. Obra 3 - Bach

5.3.1. Objetivos

En el Barroco, las composiciones se caracterizaban por marcados contrastes en el timbre, la densidad instrumental y la dinámica. Es decir, se pasaba del *pianísimo* más suave hasta el *fortísimo* más intenso, todo ello marcado por los *crescendos* y *diminuendos* que se encuentra a lo largo de estas obras. El objetivo de este experimento es elegir una obra barroca y observar como el sistema experto compone los distintos elementos ya mencionados y cual es la diferencia en cuanto a expresividad se refiere con la misma obra sin aplicar dicha expresividad.

5.3.2. Preparacion

Para la realización del experimento se va a establecer como entrada del programa el fragmento de una obra del compositor Johann Sebastian Bach llamada *"Jesu, Joy of Man's Desiring"* [37], compuesta por 2 frases a su vez formadas por 2 semifrases cada una. A continuación se muestra la distribución de éstas:

Frase 1

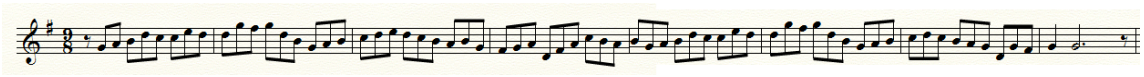

Figura 43: Obra 3 - Frase 1

Frase 2

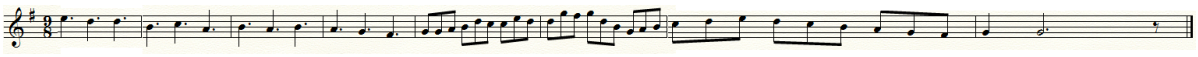

Figura 44: Obra 3 - Frase 1

Tanto la frase 1 como la frase 2 se descomponen en 2 semifrases cada una. Los motivos corresponden a cada compás de la obra.

Por tanto, la partitura quedaría como sigue:

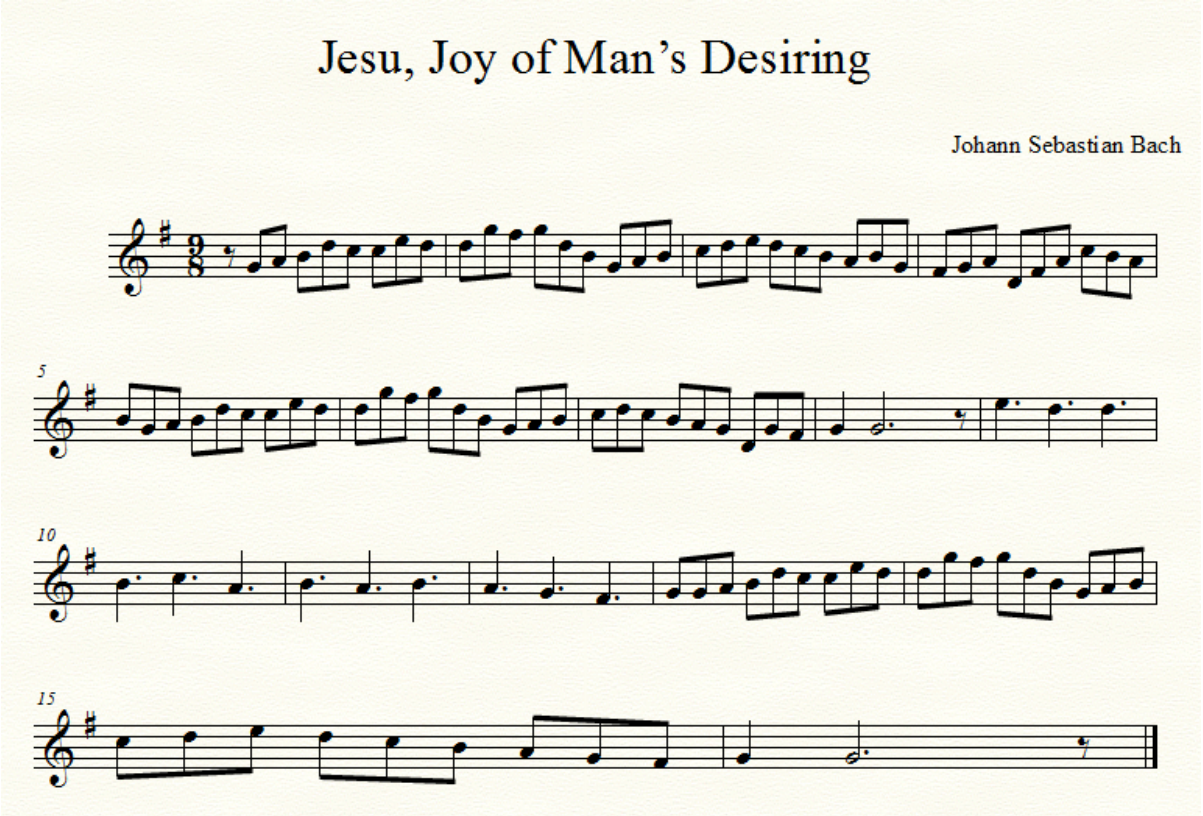

Figura 45: Obra3

5.3.3. Resultados y conclusiones

A continuación se muestra una tabla resumen indicando qué regla se ha aplicado a qué motivos:

	RI-01	RI-02	RI-03	RI-04	RI-05	RT-01	RT-02
Motivo 0				X			
Motivo 1				X	X		
Motivo 2					X		
Motivo 3				X			
Motivo 4	X			X			
Motivo 5	X		X	X	X		
Motivo 6		X			X	X	
Motivo 7		X					
Motivo 8		X			X		
Motivo 9		X			X		
Motivo 10					X		
Motivo 11							
Motivo 12	X						
Motivo 13	X			X	X		
Motivo 14					X		
Motivo 15							X

Cuadro 8: Tabla asignación regla-motivo del experimento 3

Conclusiones

Se han aplicado un total de 26 reglas sobre los 16 compases de la obra. A continuación se muestra la expresividad aplicada mediante notación musical en la obra:

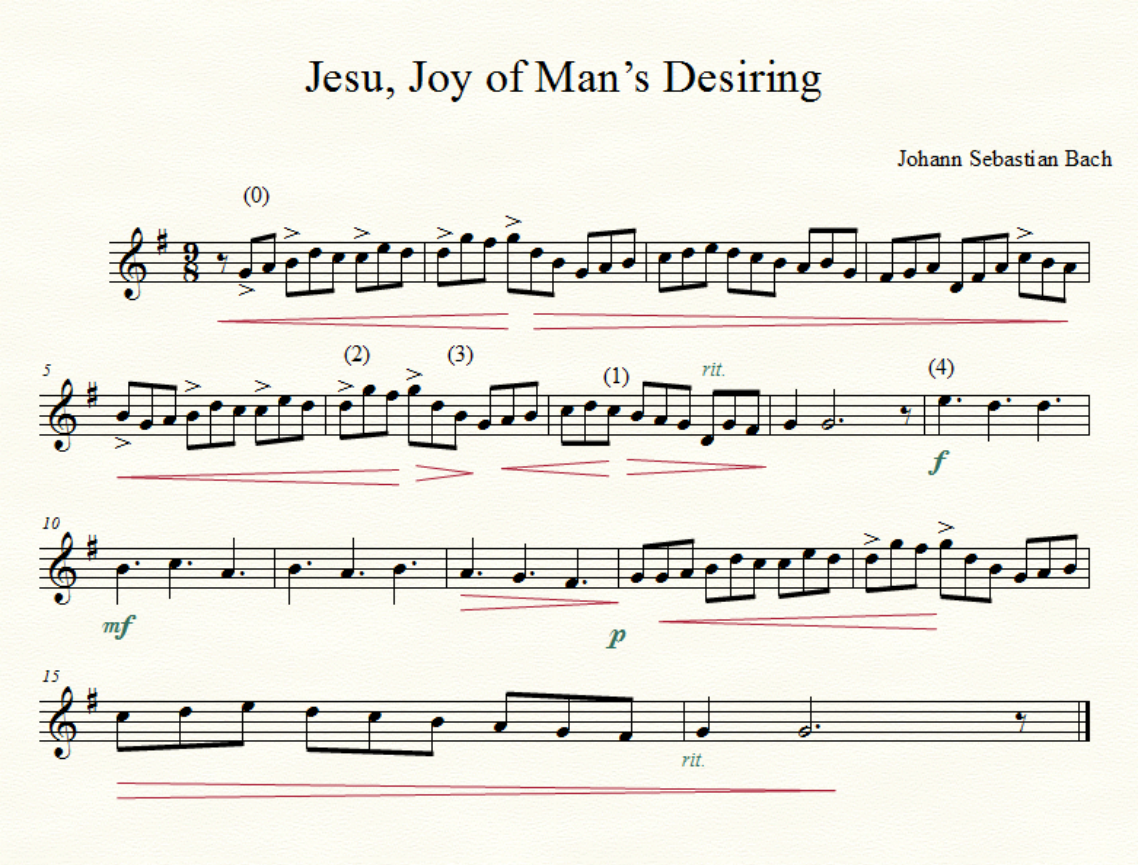

Figura 46: Obra 4 con la expresividad aplicada

En la partitura, se han representado una regla de cada tipo. A continuación se presenta la relación el nombre de la regla con su número representado en la partitura:

- Se ha aplicado la regla RI-04 sobre el motivo: 0 → (0)
- Se ha aplicado la regla RI-02 sobre el motivo: 6 → (1)
- Se ha aplicado la regla RI-01 sobre el motivo: 5 → (2)
- Se ha aplicado la regla RI-03 sobre el motivo: 5 → (3)
- Se ha aplicado la regla RI-05 sobre el motivo: 8 → (4)

En la primera frase se observa una continua acentuación de las notas y un cambio constante de intensidad, empezando con un crescendo, para luego terminar la primera semifrase con un diminuendo. La segunda semifrase, al tener la misma estructura que la primera en sus primeros motivos, de nuevo se vuelve a incrementar la intensidad hasta llegar a un ritardando suave que indica el final de la primera frase. En la segunda frase es diferente: en la primera semifrase de ésta se comienza con notas largas (negras) en la que no se diferencian acentos en las notas sino solo cambios de intensidad. De nuevo, la última semifrase es la terminación del fragmento y se repite la misma expresividad que en la primera frase con una diferencia notable al final: el ritardando más largo que indica el final de la obra.

Es posible escuchar los resultados en los siguientes ficheros:

- Obra con expresividad: *Resultados/Bach - Con Expresividad.mid*
- Obra sin expresividad: *Resultados/Bach - Sin Expresividad.mid*

5.4. Obra 4 - Vivaldi

5.4.1. Objetivos

En este experimento se pretende analizar la compatibilidad entre distintas voces y deducir si se puede o no calcular la expresividad de forma asíncrona. Es decir, ejecutar el sistema experto por separado en cuanto a voces se refiere, aplicar la expresividad en cada una de éstas de manera individual y finalmente crear el fichero de salida con todas las voces juntas. Para ello, en primer lugar se analizará las voces por separado, creando ficheros de salida individuales e identificando las partes en las que cada voz se considera ser la principal (esto es, su melodía resalte sobre las demás) y cuando no. Posteriormente, se analizará la obra conjuntamente, esto es, mezclando las voces, para analizar si el intercambio de la expresividad entre las voces es correcto. La obra elegida ha sido el Concerto Op.3 N.11 de Vivaldi [38].

5.4.2. Preparacion

Aunque esta composición contiene hasta 31 páginas, solo se va a analizar un fragmento de la obra que contiene los primeros 20 compases.

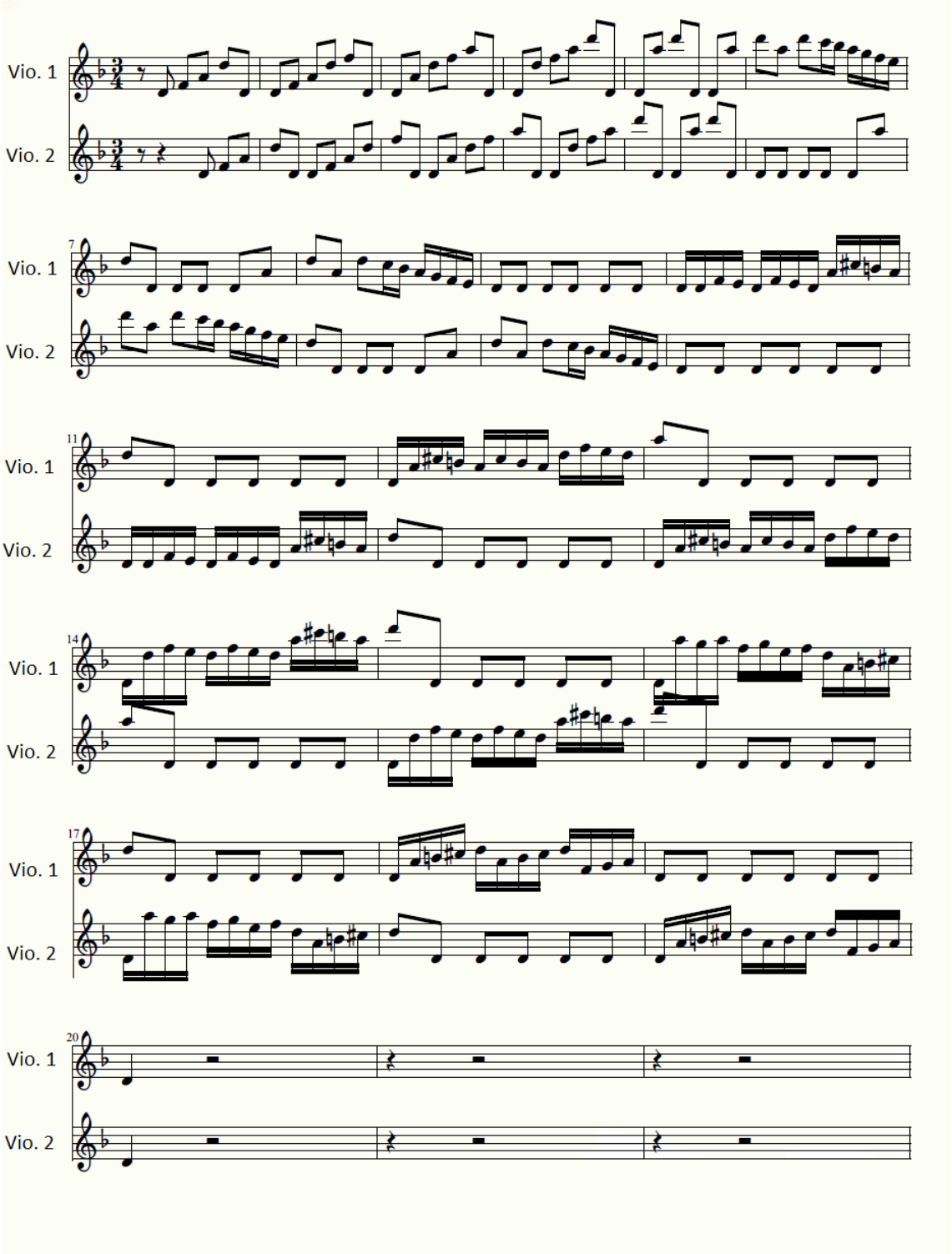

Figura 47: Obra 4 - Fragmento objeto de experimentación

En este fragmento, todo estará agrupado en una sola frase, puesto que es un fragmento de una obra más larga y por la estructura de ésta el fragmento no se puede separar. La unidad básica será el compás, identificado como cada uno de los motivos de la obra.

Dicho esto, en esta obra se pretende analizar, como se ha comentado en los objetivos del experimento, la compatibilidad entre dos voces cuya expresividad ha sido aplicada por

separado.

5.4.3. Resultados y conclusiones

A continuación se muestran los resultados de la ejecución del programa:

Reglas aplicadas a la voz 1

A continuación se muestra una tabla resumen indicando qué regla se ha aplicado a qué motivos:

	RI-01	**RI-02**	**RI-03**	**RI-04**	**RI-05**	**RT-01**	**RT-02**
Motivo 0							
Motivo 1				X	X		
Motivo 2				X	X		
Motivo 3				X	X		
Motivo 4	X			X			
Motivo 5		X		X	X		
Motivo 6	X			X			
Motivo 7				X	X		
Motivo 8				X			
Motivo 9				X	X		
Motivo 10				X			
Motivo 11				X	X		
Motivo 12				X			
Motivo 13				X	X		
Motivo 14		X		X			
Motivo 15		X		X	X		
Motivo 16		X	X	X			
Motivo 17		X		X	X		
Motivo 18			X	X	X		
Motivo 19							X

Cuadro 9: Tabla asignación regla-motivo del experimento 4 y voz 1

Reglas aplicadas a la voz 2

	RI-01	RI-02	RI-03	RI-04	RI-05	RT-01	RT-02
Motivo 0							
Motivo 1				X	X		
Motivo 2				X	X		
Motivo 3				X	X		
Motivo 4				X	X		
Motivo 5	X			X			
Motivo 6		X		X	X		
Motivo 7	X			X			
Motivo 8	X			X	X		
Motivo 9				X			
Motivo 10				X	X		
Motivo 11				X			
Motivo 12				X	X		
Motivo 13				X			
Motivo 14				X	X		
Motivo 15		X		X			
Motivo 16	X			X	X		
Motivo 17		X		X			
Motivo 18		X		X	X		
Motivo 19							X

Cuadro 10: Tabla asignación regla-motivo del experimento 4 y voz 2

Conclusiones

Esta expresividad resultaría en una partitura cuyos elementos de expresividad son similares a ésta:

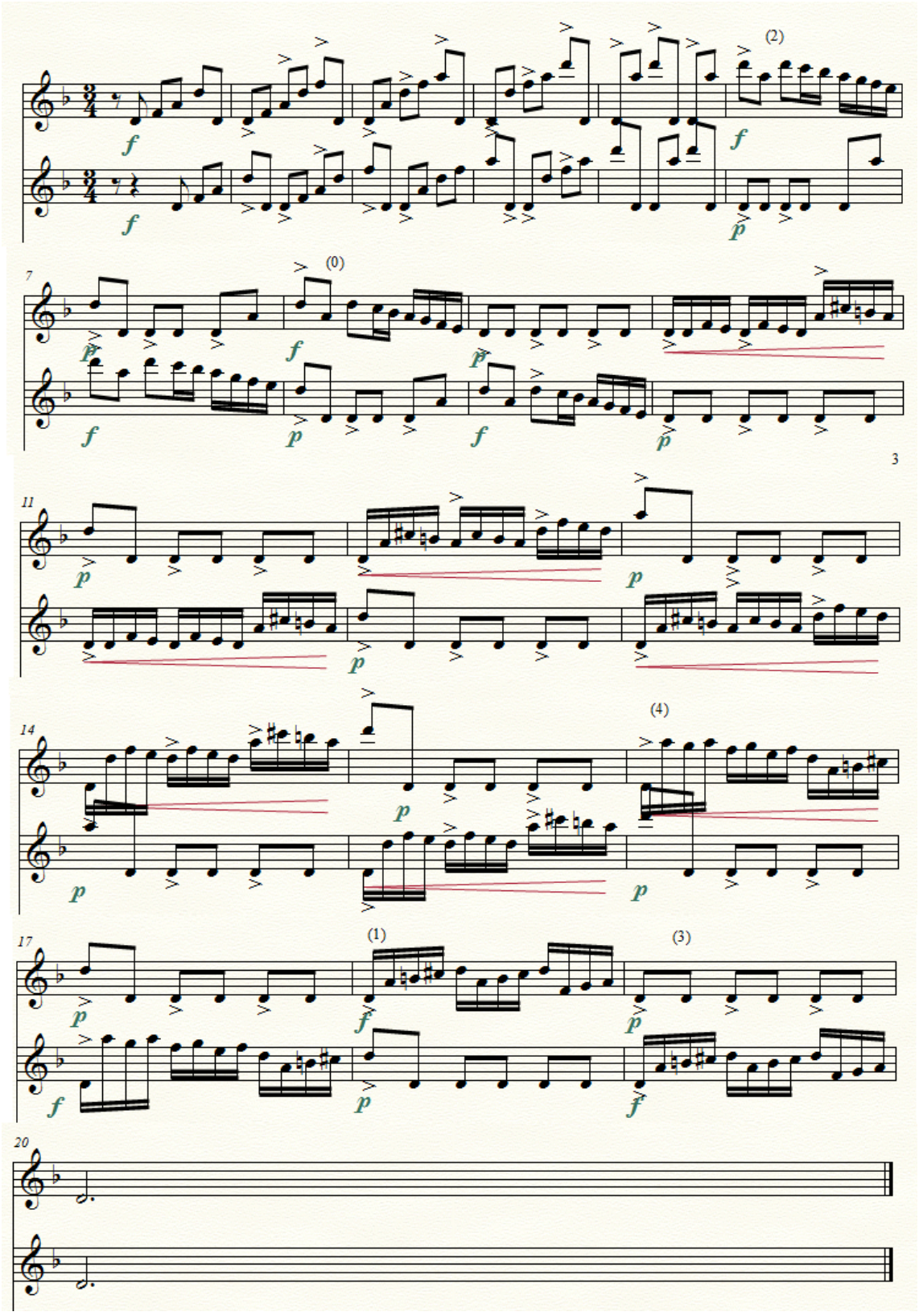

Figura 48: Obra 4 - Fragmento objeto de experimentación con la expresividad aplicada

En la partitura, se han representado una regla de cada tipo. A continuación se presenta la relación el nombre de la regla con su número representado en la partitura:

- Se ha aplicado la regla RI-04 sobre el motivo: 7 en voz 1 → (0)
- Se ha aplicado la regla RI-02 sobre el motivo: 17 en voz 1 → (1)
- Se ha aplicado la regla RI-01 sobre el motivo: 5 en voz 1 → (2)
- Se ha aplicado la regla RI-03 sobre el motivo: 18 en voz 1 → (3)
- Se ha aplicado la regla RI-05 sobre el motivo: 15 en voz 1 → (4)

En la melodía se distingue cómo ambas voces empiezan con una intensidad *mezzo-forte* para ir aumentando la intensidad hasta el motivo 5 en la voz 1. Aquí es cuando empieza la conversación entre ambas voces musicalmente hablando y es cuando se puede ver que la voz que no tiene la voz principal disminuye la intensidad, dejando a la voz principal sonar más fuerte. Este suceso se repite durante todo el fragmento y se puede apreciar especialmente por la aplicación de estas reglas:

Voz 1

Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 15
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 13
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 11
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 9
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 7
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 5
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 3
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 2
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 1

Voz 2

Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 18
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 16
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 14
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 12
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 10
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 8
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 6
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 4
Se ha aplicado la regla motivo2_mas_altura_motivo1 sobre el motivo: 2

Si se cumple la lógica que aplica a cada regla, se puede ver cómo en los motivos impares la voz 1 tendrá una intensidad mayor y en los motivos pares una intensidad menor. A su vez, la voz 2 tendrá una intensidad mayor en los motivos pares y una intensidad menor en los motivos impares. Esto hace que se consiga este efecto escuchado en la melodía.

Por último, dado que es una obra con estructuras de figuras muy repetitivas, se aplican números reglas de acentuación en las notas.

Es posible escuchar los resultados en los siguientes ficheros:

- Obra con expresividad a 2 voces: *Resultados/Vivaldi (Fragmento) - Con Expresividad - 2 voces,mid*
- Obra sin expresividad a 2 voces: *Resultados/Vivaldi (Fragmento) - Sin Expresividad - 2 voces.mid*

- Obra con expresividad solo primera voz: *Resultados/Vivaldi (Fragmento) - Con Expresividad-Primera voz.mid*
- Obra sin expresividad solo primera voz: *Resultados/Vivaldi (Fragmento) - Sin Expresividad-Primera voz.mid*
- Obra con expresividad solo segunda voz: *Resultados/Vivaldi (Fragmento) - Con Expresividad-Segunda voz.mid*
- Obra sin expresividad solo segunda voz: *Resultados/Vivaldi (Fragmento) - Sin Expresividad-Segunda voz.mid*

6. Conclusiones

Con la realización del proyecto se ha conseguido crear un sistema experto que aplique expresividad a una obra perteneciente al Barroco cuyo instrumento es el violín. Todo ello marcado por la consecución de los subobjetivos establecidos inicialmente. A continuación se exponen las conclusiones obtenidas para cada uno de ellos:

- Generación de información en formato *musicXML*. Se ha conseguido entender el formato *musicXML* así como cuales son los elementos necesarios a extraer y cuales no acorde al objetivo del proyecto. Se ha conseguido tanto crear melodías en formato *musicXML* como encontrar una amplia base de datos de partituras (www.musescore.com). Este subobjetivo correspondía con la búsqueda de información para usarla como entrada del programa.

- Definición de reglas de expresividad mediante entrevista a un experto. Se ha conseguido extraer reglas esenciales para el violín en la época del Barroco a través de entrevistas a un experto. En concreto se han extraído 5 reglas que modifican la intensidad y 2 que modifican el tiempo de la obra.

- Creación de reglas de expresividad mediante un motor de reglas. Se ha conseguido definir unas reglas de expresividad relacionadas con obras para violín pertenecientes al Barroco mediante entrevistas a un experto y a su vez haberlas programado en *CLISP*. Estas reglas han podido ser leídas mediante el motor de reglas *JESS*, el cual ha sido integrado en la aplicación. Por último, se ha creado el mecanismo necesario para que la salida del motor de reglas pueda ser extraída fácilmente. Este mecanismo consiste en la creación de un conector que almacene el resultado de la red *Rete* y que pueda ser consultado por las clases *JAVA* del sistema. En la sección 6.1 se exponen algunas mejoras/extensiones para este subobjetivo.

- Modificación de la información acorde a las reglas de expresividad encontradas. Mediante la salida del motor de reglas, se ha conseguido leer ésta y crear así el conjunto de notas musicales finales que contengan la expresividad de la obra. Se ha conseguido aplicar efectos propios del violín, así como variaciones en la intensidad y tiempo de la obra.

- Exportación de la información en formato *MIDI*. Se ha conseguido crear un reproductor que use las clases propias de *JAVA* que posibilitan la creación de un fichero *MIDI* a partir de la lista de notas musicales interpretadas por el motor de reglas. Este reproductor es capaz de escribir varias voces a la vez, lo que resulta en melodías polifónicas en lugar de monofónicas (una sola voz).

- Evaluación de las melodías en formato *MIDI*. Se ha realizado una sección dedicada a este subobjetivo en el que se ha comprobado la asignación de cada regla a cada motivo, verificando si era correcta o no teniendo en cuenta la acción-efecto de cada regla usando como acción el motivo aplicado y como efecto las modificaciones hechas sobre éste. Se puede concluir que las reglas han sido correctamente aplicadas a cada obra.

6.1. Líneas futuras

Durante la realización del trabajo, se han encontrado algunos puntos de mejora que pueden servir como líneas futuras para la continuación del trabajo. Estos puntos están expuestos a continuación:

- Ampliación de obras de otras épocas: Consistiría en crear reglas de expresividad propias de otras épocas tales como el Renacimiento o la época Contemporánea. Esta parte sería una extensión del sistema experto programado con *JESS*.

- Ampliación de obras de otras familias de instrumentos: Consistiría en crear reglas de expresividad para otras familias de instrumentos como por ejemplo el viento, definiendo así instrumentos tales como el clarinete, oboe, fagot, etc. Esta parte sería una extensión del sistema experto programado con *JESS*.

- Creación de un método automático para elegir las reglas de expresividad. Se trataría de crear un mecanismo que a partir de una obra dada, elegir unas reglas de expresividad y descartar otras en función de ciertos elementos como podrían ser:

  - Autor de la obra: Permitiría saber la época a la que pertenece
  - Instrumento(s) de la obra
  - Técnica supervisada para clasificar obras en diferentes categorías: Dada una relación entre época de la obra y las reglas de expresividad aplicadas, se trataría de crear un modelo entrenado con diferentes obras de distintas épocas para después clasificar obras nuevas y así saber qué reglas de expresividad aplicar y cuales no. Así, no haría falta analizar cada obra específica.

  Esta mejora debería ir en un paquete nuevo ya que sus características no pertenecen a ningún módulo ya creado.

- Definición avanzada de las frases, semifrases y motivos de la obra. Se trataría de crear un modelo el cual encuentre las frases, semifrases y motivos de la obra de una mantera más avanzada que la actual. Habría varias posibilidades:

  - Técnicas de aprendizaje supervisado: Se trataría de crear un modelo entrenado por varias obras de varios autores. La categorización se haría en base a la época y al tipo de composición, ya que en general, éstos tendían a cumplir estructuras específicas y bien documentadas. Así, cuando se quisiera analizar una nueva obra, dado el modelo que conoce las frases, semifrases y motivos de las obras que pertenecen a su misma categoría, se podrían definir estos elementos que cumplan patrones similares (estructura de los motivos, altura de éstos, variación de tonalidades, etc).
  - Creación de reglas: Se trataría de crear un sistema experto paralelo que defina la estructura que seguían las composiciones de cada época a analizar y que sea éste quien defina las frases, semifrases y motivos.

  En resumen, este punto se basa en que los autores de una misma época compartían las mismas estructuras en sus composiciones (óperas, sinfonías, etc). Existe mucha documentación al respecto y hay asignaturas en conservatorios que se dedican solo a ésto (Análisis y Composición).

- Ampliar el repertorio de efectos del Barroco para el violín. En el proyecto solo se ha definido un efecto, el de acentuar las notas, podrían implementarse muchos más como por ejemplo el *vibrato* [28], dobles cuerdas (efecto producido cuando se toca mas de una cuerda a la vez), etc.

- Ampliar la expresividad musical que resulta de la comunicación entre dos o más voces, esto es, no analizar cada voz por separado, sino aplicar reglas de expresividad que estén inter-conectadas entre los distintos instrumentos. Se trataría de crear nuevos hechos y reglas en los que poder relacionar los distintos elementos de la fraseología musical entre las distintas voces.

7. Planificación

En este capítulo se expone la metodología de desarrollo así como la planificación seguida para la realización del proyecto, explicado con un diagrama de Gantt. También se muestra una estimación de costes del proyecto con COCOMO [34].

7.1. Metodología de desarrollo y su planificación

La metodología de desarrollo ha sido iterativa e incremental, ya que el desarrollo de un nuevo módulo dependía de la implementación del anterior y debido al aprendizaje y profundización en la temática del proyecto durante su desarrollo, el diseño inicial de cada módulo no era suficiente para cubrir toda la funcionalidad.
El proyecto comenzó el 18 de junio de 2014 y terminó el 28 de diciembre de 2014, con una duración de 118 días sin contar festivos ni fines de semana. Las tareas han sido desarrolladas una tras otra excepto la documentación, que se ha ido desarrollando en paralelo a todas las demás excepto la investigación. A continuación se muestra el diagrama de Gantt del proyecto, mostrando cada tarea relacionada con la investigación, diseño, desarrollo, experimentación y documentación del mismo.

Nombre de tarea	Duración	Comienzo	Fin
Proyecto	**118 días**	**mié 18/06/14**	**vie 28/11/14**
Investigación	**14 días**	**mié 18/06/14**	**mar 08/07/14**
Estado del arte	4 días	mié 18/06/14	mar 24/06/14
Tecnologías a usar	2 días	mar 24/06/14	jue 26/06/14
Aprendizaje del uso de las tecnologías	8 días	jue 26/06/14	mar 08/07/14
Diseño	**8 días**	**mar 08/07/14**	**vie 18/07/14**
Arquitectura del sistema	3 días	mar 08/07/14	vie 11/07/14
Arquitectura de cada módulo del sistema	5 días	vie 11/07/14	vie 18/07/14
Desarrollo	**81 días**	**vie 18/07/14**	**lun 10/11/14**
Creación del parser musicXML	15 días	vie 18/07/14	vie 08/08/14
Definición de reglas de expresividad	20 días	vie 08/08/14	vie 05/09/14
Integración de JESS en el sistema	28 días	vie 05/09/14	mié 15/10/14
Reproductor MIDI	18 días	mié 15/10/14	lun 10/11/14
Experimentación	**10 días**	**lun 10/11/14**	**lun 24/11/14**
Definición de casos	5 días	lun 10/11/14	lun 17/11/14
Ejecución y conclusiones	5 días	lun 17/11/14	lun 24/11/14
Documentación	101 días	vie 11/07/14	vie 28/11/14

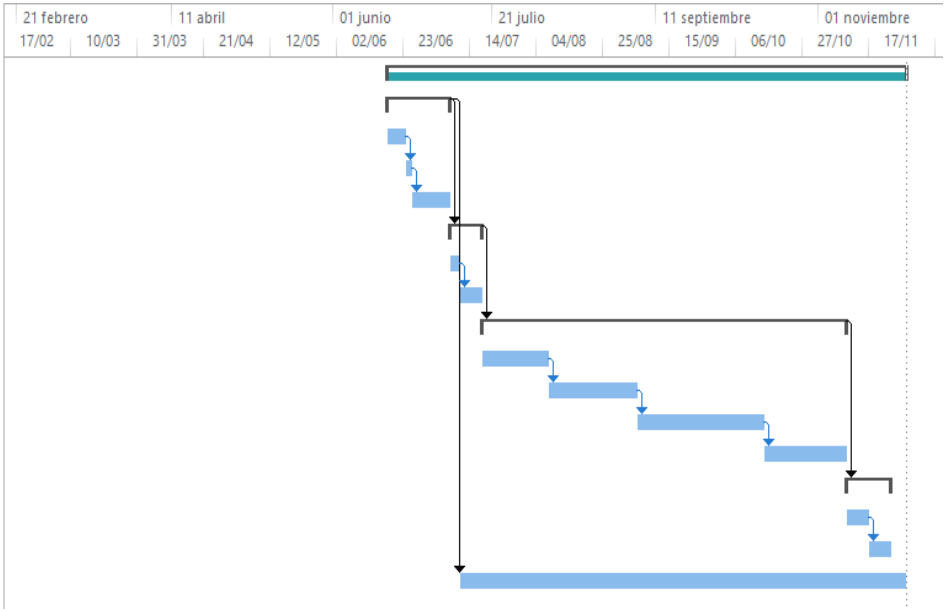

Figura 49: Diagrama de Gantt mostrando la planificación del proyecto

Se puede observar como la integración del sistema experto ha sido la parte que más tiempo ha llevado, con un total de 48 días (definición de reglas de expresividad + adaptación de *JESS* en el sistema).

7.2. Estimación del coste del proyecto

A continuación, se muestra la estimación del coste del proyecto usando la metodología COCOMO.
Esta metodología utiliza tres submodelos para estimar los costes. Estos tres submodelos se diferencian por su nivel de detalle y aproximación en los costes. Éstos son:

- Modelo básico: Se utiliza para obtener una primera aproximación del esfuerzo.
- Modelo intermedio: Incrementa la precisión de los costes añadiendo indicadores que tienen en cuenta el entorno de trabajo.
- Modelo detallado: Los indicadores de esfuerzo depende de cada fase, es decir, su valor puede ser variable dependiendo en si se aplica en una fase u otra. Además, el producto se divide en tres niveles: módulo, subsistema y sistema. Dicha división se utiliza para clasificar los indicadores, siendo los que presentan gran variación a bajo nivel los del módulo, pocas variaciones los del subsistema y el restos los del sistema [35].

Después, cada modelo, se divide en *modos* que representan el tipo de proyecto, estos son [34]:

- Modo orgánico: Un pequeño grupo de programadores experimentados desarrollan software en un entorno familiar. El tamaño del software varía desde unos pocos miles de líneas (tamaño pequeño) a unas decenas de miles (medio).
- Modo semilibre: Corresponde a un esquema intermedio entre el orgánico y el rígido; el grupo de desarrollo puede incluir una mezcla de personas experimentadas y no experimentadas.
- Modo rígido: El proyecto tiene fuertes restricciones, que pueden estar relacionadas con la funcionalidad y/o pueden ser técnicas. El problema a resolver es único y es difícil basarse en la experiencia, puesto que puede no haberla.

Una vez analizados los modelos y los modos, se ha decido situar este proyecto en el modelo básico y modo orgánico, ya que el código posee alrededor de 2600 líneas (sin contar la programación de las reglas de expresividad musical) y el entorno de desarrollo ha sido estable y sin presión por parte de los usuarios.
Por tanto, dada la siguiente tabla de constantes para la estimación de costes para el modelo orgánico:

MODO	**a**	**b**	**c**	**d**
Orgánico	2.40	1.05	2.50	0.38
Semi - Orgánico	3.00	1.12	2.50	0.35
Empotrado	3.60	1.20	2.50	0.32

Cuadro 11: Tabla constantes para modelo orgánico COCOMO

Y dadas las siguientes fórmulas:

- Esfuerzo por persona/mes: $E = a * KLOC^b$ donde KLOC es el número de líneas de código.
- Tiempo de desarrollo en meses: $D = c * E^d$
- Número de personas necesarias: $P = E/D$

Se estima el numero total de recursos necesarios para el proyecto:

$$E = 2{,}40 * 2{,}6^{1}{,}05 = 6{,}54 \text{ meses/hombre } D = 2{,}50 * 6{,}54^{0}{,}38 = 5{,}10 \text{ meses}$$

Finalmente, el número total de recursos necesarios para el desarrollo del proyecto es:

$$P = E/D = 6{,}54/5{,}10 = 1,28 \approx \textbf{1 persona}.$$

8. Presupuesto

En esta sección se detalla el presupuesto del proyecto basado en la estimación de costes realizada en la sección anterior (1 persona como recurso humano).

8.1. Gastos de personal

Para realizar el coste de personal del proyecto, se calcularán los salarios para 1 persona con dos distintos roles: el de analista y el de programador junior.
El analista se encargará de identificar los requisitos necesarios para el desarrollo del sistema así como un diseño de la arquitectura y la documentación del proyecto.
El programador junior se encargará de configurar el entorno de de desarrollo, implementar el programa diseñado y el conjunto de pruebas de experimentación.
Por último, las horas diarias realizadas para la fase de Investigación son de 8 horas, y para Diseño, Desarrollo y Experimentación son 6 horas ya que las 2 restantes corresponden a la documentación realizada en paralelo. En cuanto a la documentación, se distingue entre la fase 1, en la que la documentación ha sido realizada en paralelo a las demás tareas, y la fase 2, en las que solo se ha documentado, dedicando para ello 8 horas.
Por tanto, los costes de personal son los siguientes:

Fase	**Rol**	**Salario (€/hora)**	**Horas/día**	**Días**	**Costes(€)**
Investigación	Analista	40	8	14	4480
Diseño	Analista	40	6	8	1920
Desarrollo	Programador Junior	30	6	81	14580
Experimentación	Programador Junior	30	6	10	1800
Documentación fase 1	Analista	40	2	97	7760
Documentación fase 2	Analista	40	8	4	1280
				TOTAL	31820

Cuadro 12: Tabla con costes de personal

8.2. Equipos informáticos

A continuación se muestran los equipos informáticos usados para la realización del proyecto:

Equipo	**Coste(€)/unidad**	**Unidades**	**Coste Total (€)**	**Periodo de amortización (meses)**	**Duración del proyecto(meses)**	**Coste Final (€)**
HP Pavilion 15-p100ns	744,15	1	744,15	24	7	217,04
					TOTAL	217,04

Cuadro 13: Tabla con costes de equipos informáticos

8.3. Licencias

A continuación se muestran las licencias usadas para la realización del proyecto:

Software	Coste (€)
Finale Music	106,20
TOTAL	106,20

Cuadro 14: Tabla con costes de licencias software

8.4. Material fungible

A continuación se muestran los costes relacionados con material fungible:

Concepto	Precio (€)
Papel	18
Material de escritura	10
Disco de almacenamiento externo	25
TOTAL	53

Cuadro 15: Tabla con costes de material fungible

8.5. Presupuesto total

Finalmente, el presupuesto total del proyecto es:

Tipo de coste	Precio (€)
Personal	31820
Equipos informáticos	217,04
Licencias	106,20
Material fungible	53
TOTAL SIN IVA	32196,24
TOTAL CON IVA (21 %	38957,45

Cuadro 16: Tabla con el total de costes

Por tanto, el presupuesto del proyecto asciende a un total de 38957,45 €(TREINTA Y OCHO MIL NUEVECIENTOS CINCUENTA Y SIETE CON CUARENTA Y CINCO EUROS).

Anexo 1: Notación de figuras musicales

Nota	Silencio	Nombre de la figura	Duración relativa
		Redonda	1
		Blanca	1/2
		Negra	1/4
		Corchea	1/8
		Semicorchea	1/16
		Fusa	1/32
		Semifusa	1/64

Cuadro 17: Tabla con la relación de figura con su silencio, nombre y duración relativa

Figura	Codificación
Redonda	w
Blanca	h
Negra	q
Corchea	i
Semicorchea	s
Fusa	t
Semifusa	x

Cuadro 18: Representación de las figuras en las reglas

Figura	Codificación
Do	C
Re	D
Mi	E
Fa	F
Sol	G
La	A
Si	B

Cuadro 19: Representación del nombre de las notas en las reglas

Manual de usuario

En esta sección se detalla cómo ejecutar el programa así como los requisitos para poder hacerlo.
El programa puede ejecutarse tanto en entornos de Windows XP o superior. También es obligatorio tener *JAVA* instalado. La versión de ésta ha de ser JDK 1.6 o superior.

Nota: El manual se realiza desde el directorio "C:/PFC"

El comando para ejecutar el programa tanto en Windows como en Linux es el siguiente:

```
java -cp ./extlib/jess.jar;./extlib/jdom.jar;build/classes SistemaExperto
```

Ahora se muestra el ejemplo de una ejecución, explicando cada paso detalladamente. La ejecución se realizará en un PC con windows 8.

- Ejecución del programa: Se trata de ejecutar el comando anterior, haciendo referencia a la librería *JESS* (el motor de reglas) y *JDOM* librería para el parseo de los ficheros en formato *musicXML*

Figura 50: Ejecución del programa

- Especifique la ruta del fichero de la obra a leer. Por ejemplo, dado el siguiente directorio:

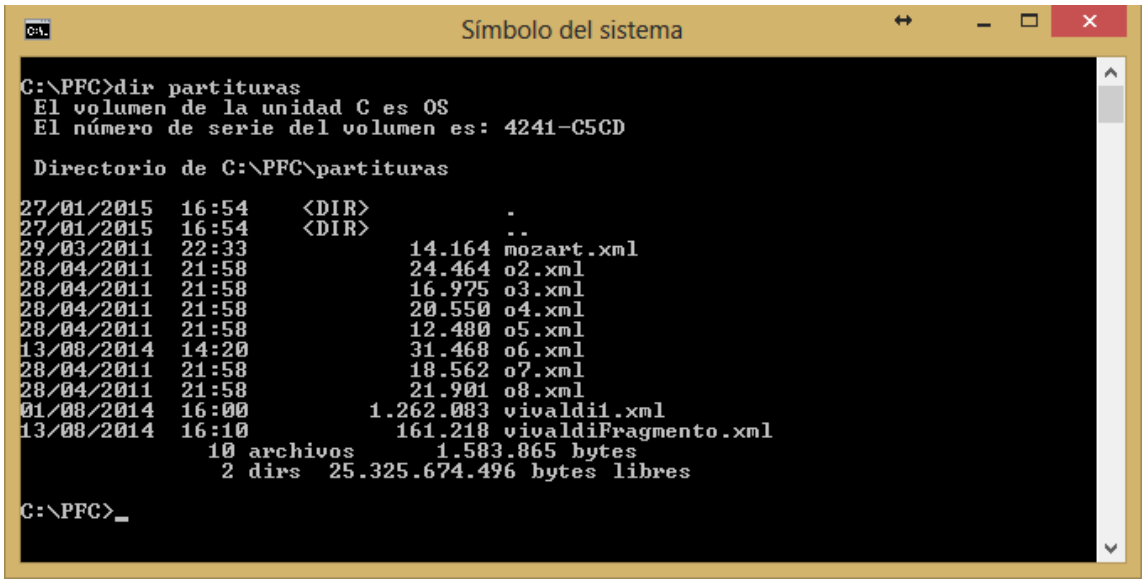

Figura 51: Partituras para el manual de usuario

Se quiere leer la obra "mozart.xml", por tanto la entrada sería

```
partituras/mozart.xml
```

- Especifique el número de voces a extraer. Sirve para especificar el número de voces de la partitura que quieren ser analizadas. Ha de ser un valor numérico y debe ser menor o igual al número de voces de la obra.

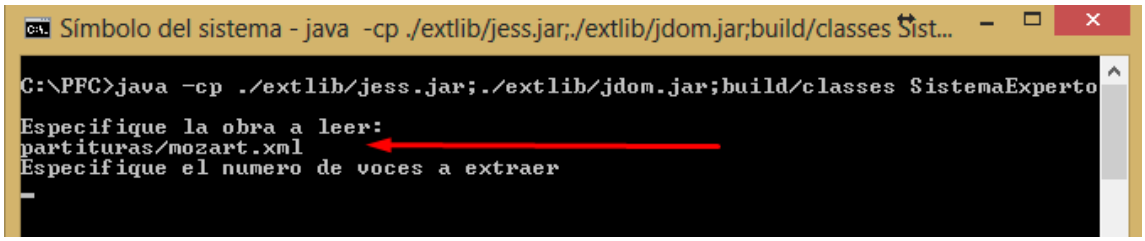

Figura 52: Partituras para el manual de usuario

Para el ejemplo, se seleccionará una voz.

- Especifique el tempo de la obra. Sirve para definir la velocidad a la que la obra se va a reproducir. Los valores aceptados son entre entre 20 (el tempo musical más lento) y 240 (el tempo musical más rápido).

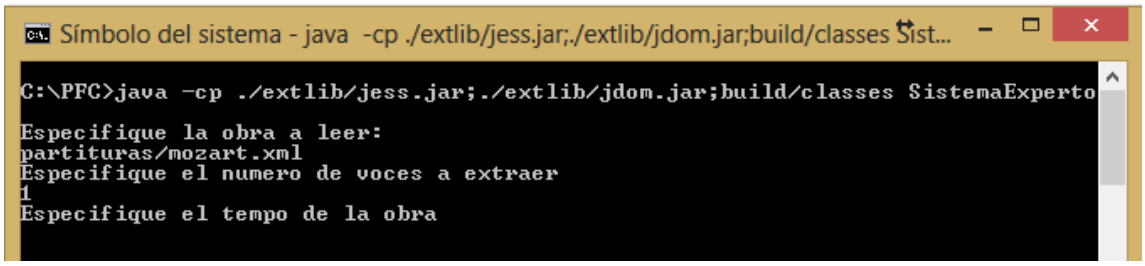

Figura 53: Manual de usuario. Especificar el tempo de la obra

Para el ejemplo, se seleccionará 120.

- Especifique el fichero de salida (opcional). Sirve para definir la ruta y nombre en el que se escribirá la melodía, generando el fichero *MIDI*. La ruta ha de existir. Si no se define ningún valor, la melodía será reproducida tras la aplicación de las reglas de expresividad.

Figura 54: Manual de usuario. Especificar el fichero de salida

Para el ejemplo, se generará el fichero de salida “melodias/mozart.mid”

- Especifique el fichero de metadata definiendo las partes de la obra (opcional). Sirve para definir la estructura de la melodía.

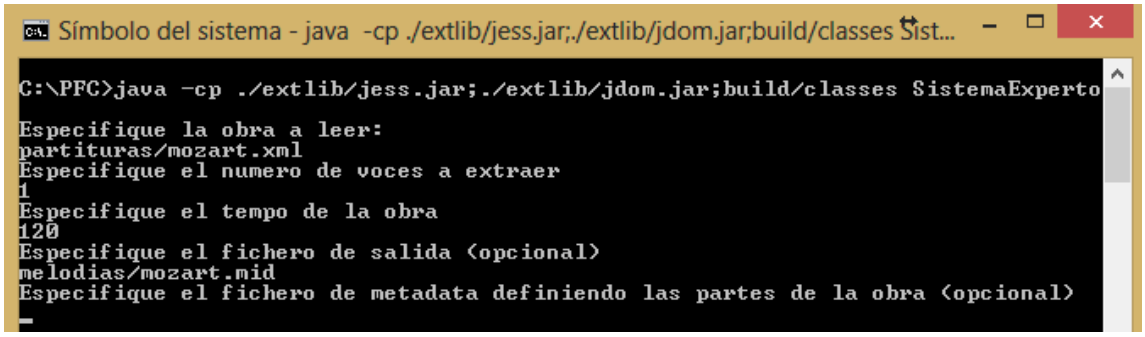

Figura 55: Manual de usuario. Especificar el fichero de metadata

La definición o no de la ruta de este fichero hace que el programa calcule la estructura de la obra de dos formas distintas:

- Si se define la ruta del fichero de metadata, el sistema mapea la información de este fichero con la partitura definida de entrada. Este fichero de metadata tiene el siguiente formato:
  - Es un fichero en formato CSV.
  - Las columnas de este fichero son:
    1. Elemento: Puede ser "frase", "semifrase" o "motivo".
    2. CompasInicio: Es el número de compás del inicio del elemento.
    3. CompasFin: Es el número de compás del fin del elemento.

    De esta forma, el sistema lo lee y automáticamente genera las relaciones explicadas en la arquitectura del sistema entre las instancias de las clases *Frase*, *Semifrase* y *Motivo*.

  Este sería un ejemplo que serviría como fichero de metadata:

```
frase,1,8
semifrase,1,4
motivo,1,1
motivo,2,2
motivo,3,4
semifrase,5,8
motivo,5,5
motivo,6,6
motivo,7,8
frase,9,16
semifrase,9,12
motivo,9,9
motivo,10,10
motivo,11,12
semifrase,13,16
motivo,13,13
motivo,14,14
motivo,15,16
```

Figura 56: Manual de usuario. Ejemplo de fichero de metadata

- Si no se define ningún valor, el sistema calculará automáticamente la estructura de la obra. Para obras complejas, dado que no era el objetivo del proyecto, se recomienda definir un fichero de metadata para una correcta aplicación de expresividad.

Para el ejemplo, se dejará esta entrada vacía para que el sistema defina la estructura de la obra automáticamente.

- Especifique si quiere aplicar la expresividad de la obra. Sirve para experimentación, comparando la salida del programa aplicando y sin aplicar expresividad. Los valores aceptados son *true* o *false* respectivamente.

  Para el ejemplo, se elegirá que sí aplique expresividad.

  Por tanto, la entrada del programa quedaría así:

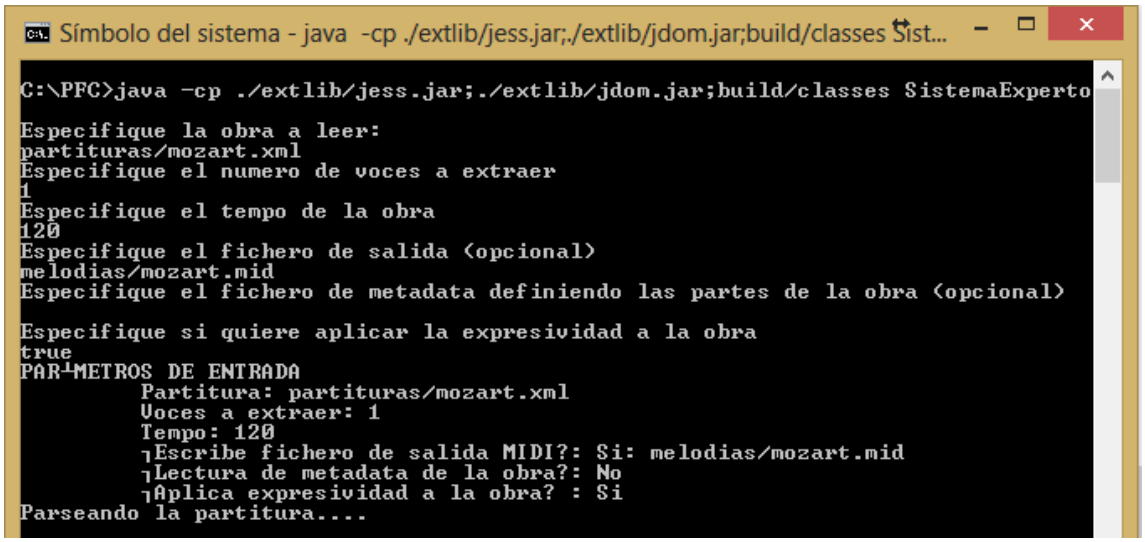

Figura 57: Manual de usuario. Entrada completa del programa

Finalmente, una vez que el sistema experto termina su ejecución, ya es posible reproducir el fichero *MIDI* generado o bien escuchar la melodía en el momento, dependiendo de si se ha definido un fichero de salida o no. Así, se obtendría la salida final del programa.

Referencias

[1] *Significado de expresivo. Real Academia Española. Disponible [Internet]:* http://lema.rae.es/drae/?val=expresivo

[2] *Conceptos básicos para el inicio en el análisis. Isabel Delgado. Disponible [Internet]:* http://www.cpmjaen.es/node/1159

[3] *Finale. Makemusic corporation 2012. Disponible [Internet]:* http://www.finalemusic.com

[4] *Werner Schweer y otros. 2002. MuseScore. Disponible [Internet]:* http://www.musescore.org

[5] *MusicXML. Makemusic corporation 2012. Disponible [Internet]:* http://www.makemusic.com/musicxml/

[6] *Wikipedia. Definición de melodía. Disponible [Internet]:* http://es.wikipedia.org/wiki/Melodia

[7] *Wikipedia. Definición de música. Disponible [Internet]:* http://es.wikipedia.org/wiki/Musica

[8] *Music and Artificial Intelligence.. Chris Dobrian. 1993. Disponible [Internet]:* http://music.arts.uci.edu/dobrian/CD.music.ai.htm

[9] *Music and Artificial Intelligence. Blog Think Big. 2014. Disponible [Internet]:* http://blogthinkbig.com/inteligencia-artificial-musica/

[10] *Music and Artificial Intelligence. Acens. 2005. Disponible [Internet]:* http://www.acens.com/articulos/inteligencia-artificial-el-adn-de-la-musica/

[11] *Wikipedia. Audio Digital. Disponible [Internet]:* http://es.wikipedia.org/wiki/Audio_digital

[12] *Wikipedia. Audio Digital. Disponible [Internet]:* http://es.wikipedia.org/wiki/Reproduccion_y_grabacion_de_sonido

[13] *Wikipedia. Artificial Intelligence in music. Disponible [Internet]:* http://en.wikipedia.org/wiki/Music_and_artificial_intelligence

[14] *Experiments in Musical Intelligence. David Cope. 2015. Disponible [Internet]:* http://artsites.ucsc.edu/faculty/cope/experiments.htm

[15] *Improvisation without representation: Artificial Intelligence and Music. Linson, Adam; Dobbyn, Chris and Laney, Robin. 2012. Disponible [Internet]:* http://oro.open.ac.uk/33355/

[16] *From Composition to Expressive Performance. Ramón Lopez de Mantaras y Josep Lluis Arcos. 2002. Disponible [Internet]:* http://www.iiia.csic.es/~mantaras/AIMag23-03-006.pdf

[17] *Toward an Expert System for Expressive Musical Performance. Margaret L. Johnson, Stanford University. 1991. Disponible [Internet]:* http://www.researchgate.net/publication/2954013_Toward_an_expert_system_for_expressive_musical_performance

[18] *Expressivity and musical performance: Practice strategies for pianists. Alfonso Benetti Jr. 2013. Disponible [Internet]:* http://www.cmpcp.ac.uk/PSN2/PSN2013_Benetti.pdf

[19] *Wikipedia. El clave bien temperado. Disponible [Internet]:* http://es.wikipedia.org/wiki/El_clave_bien_temperado

[20] *Generating Musical Performances with Director Musices [Internet]:* http://www.speech.kth.se/prod/publications/files/1770.pdf

[21] *Symbolic and audio processing to change the expressive intention of a recorded music performance. Sergio Canazza et al. 1999. Disponible [Internet]:* http://www.iet.ntnu.no/groups/akustikk/meetings/DAFx99/canazza.pdf

[22] *Jess Rule Engine. Disponible [Internet]:* http://www.jessrules.com/

[23] *CLIPS. Disponible [Internet]:* http://clipsrules.sourceforge.net/

[24] *Jess Information. Disponible [Internet]:* http://herzberg.ca.sandia.gov/

[25] *Jess Working Memory. Disponible [Internet]:* http://www.jessrules.com/doc/70/memory.html

[26] *Nombre de las notas, Tonos y semitonos. Disponible [Internet]:* http://comotocar-guitarra.blogspot.com.es/2011/12/nombre-de-las-notas.html

[27] *Las notas del teclado. Disponible [Internet]:* http://mariomusica.com/piano/notas-teclado.html

[28] *Generation of Vibrato in Expressive Performances in MIDI. Disponible [Internet]:* http://www-classes.usc.edu/engr/ise/599muscog/2004/projects/yang/

[29] *MIDI Protocol. Disponible [Internet]:* http://www.midi.org/

[30] *musicXML Public License. Disponible [Internet]:* http://www.musicxml.com/dtds/license.html

[31] *JDOM library. Disponible [Internet]:* http://www.jdom.org/

[32] *Java Sound API. Disponible [Internet]:* http://www.oracle.com/technetwork/java/index-jsp-140234.html

[33] *Algoritmo RETE. Disponible [Internet]:* http://es.wikipedia.org/wiki/Algoritmo_Rete

[34] *Definición COCOMO. Disponible [Internet]:* http://es.wikipedia.org/wiki/COCOMO

[35] *Modelo COCOMO. Disponible [Internet]:* http://es.slideshare.net/johannamartinez28/modelo-cocomo-1

[36] *6 German Dances, K.606. W.A Mozart. Disponible [Internet]:* http://www.classicalarchives.com/work/157131.html

[37] *Jesu, Joy of Man's Desiring. J.S. Bach. Disponible [Internet]:* http://en.wikipedia.org/wiki/Jesu,_Joy_of_Man%27s_Desiring

[38] *L'estro armonico. Antonio Vivaldi. Disponible [Internet]:* http://es.wikipedia.org/wiki/L%27Estro_Armonico

[39] *Online JESS Documentation. Disponible [Internet]:* http://www.jessrules.com/jess/docs/index.shtml

[40] *JBoss Rule Manual. Disponible [Internet]:* https://access.redhat.com/documentation/en-US/JBoss_Enterprise_SOA_Platform/4.2/html-single/JBoss_Rules_Manual/index.html

[41] *SAXEX. A Case-Based Reasoning system for generating expressive performances. Josep Lluis Arcos, Lopez de Mantaras R y Serra X. Disponible [Internet]:* http://www.iiia.csic.es/Projects/music/Saxex.html

[42] *Spectral modeling synthesis definition. Disponible [Internet]:* http://en.wikipedia.org/wiki/Spectral_modeling_synthesis