Universidad Carlos III de Madrid

Escuela Politécnica Superior

SISTEMA EXPERTO DE COMPOSICIÓN DE DICTADOS MUSICALES

Proyecto Fin de Carrera

Ingeniería Técnica de Telecomunicación

Sonido e Imagen

Autora: Lucía Holguín Labajo

Tutor: Julio Villena Román

Cotutora: Raquel M. Crespo García

Septiembre de 2009

Título: Sistema Experto de Composición de Dictados Musicales

Autora: Lucía Holguín Labajo

Tutor: Julio Villena Román

Cotutora: Raquel M. Crespo García

EL TRIBUNAL

Presidente: José Jesús García Rueda

Secretario: Isaac Seoane Pujol

Vocal: José Miguel Leiva Murillo

Realizado el acto de defensa del Proyecto Fin de Carrera el día 10 de Septiembre de 2009 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

Fdo: Presidente Fdo: Secretario Fdo: Vocal

Agradecimientos

Quiero agradecer la realización de este proyecto y todo el trabajo y esfuerzo anterior a mis padres y hermano, a mi familia, a Héctor, a la familia Sonimágica, al resto de mis amigos, y a todos aquellos que día a día estáis ahí o que algún día estuvisteis.

A mis maestros, mis profesores, y a mis tutores, por enseñarme y ayudarme en todo lo necesario a lo largo de mi educación.

Resumen

El presente proyecto consiste en el diseño y la implementación de un sistema experto de composición de dictados musicales cuyo principal objetivo es desarrollar un sistema capaz de simular a un experto en música que compone dictados. Este sistema está formado por varios módulos que realizan las funciones de decisión de características, construcción y presentación de resultados al usuario. Un primer módulo en el que el usuario define sus preferencias respecto al nivel y tipo de dictado que desea. Un segundo módulo en el que el sistema decidirá los parámetros que lo componen. En este módulo se generará realmente el dictado. En el tercer módulo el sistema tendrá que dar forma a esos parámetros y crear la partitura. A partir de esta partitura se creará un archivo de audio del dictado. Una cuarta fase presentará auditivamente y visualmente los resultados, permitiendo elegir al usuario que función realizar primero.

Índice

Índice de tablas

Índice de figuras

1. INTRODUCCIÓN

1.1. Motivación del proyecto

En los estudios oficiales de Música, se imparten distintas materias. Una de las asignaturas que se imparten los primeros años es la asignatura de Solfeo. Esta asignatura es la base para los conocimientos que se irán adquiriendo a lo largo de los años y por ello comprende varias disciplinas, entre las que están los dictados musicales. Un dictado musical es un ejercicio auditivo en el cual, el estudiante debe transcribir la música que está escuchando. Debe intentar identificar las notas, los tiempos y los ritmos correctamente.

Normalmente, para el estudio particular de la mayoría de las disciplinas, el alumno puede realizarlo en solitario, con la ayuda de los libros u otros materiales. Sin embargo, cuando el alumno tiene que practicar los dictados, necesita a otra persona que toque la pieza que él tendrá que transcribir. Aquí es donde surge el problema ya que no solo tendría que contar con la ayuda de otra persona sino que además tendría que ser alguien que tuviera los conocimientos musicales necesarios para tocar la pieza, lo que hace más difícil practicar esta parte de la asignatura fuera de las horas lectivas.

Es por esto que surge la idea de crear un sistema inteligente que genere dictados musicales de manera que el alumno pueda estudiar por su cuenta.

1.2. Objetivos del Proyecto

El **objetivo** de este proyecto es desarrollar un sistema experto que sea capaz de componer dictados musicales de diferentes niveles de dificultad, a partir de unos parámetros de configuración, y permitir su reproducción y visualización para poder corregirlos, además de generar un fichero MIDI que se pueda reproducir con programas convencionales.

Para conseguir crear un sistema basado en los sistemas expertos se utiliza el concepto de la Inteligencia Artificial, es decir, el estudio de la realidad expresándola de tal manera que pueda ser manipulada por una máquina inteligente.

Un sistema experto se define como un sistema que simula a los expertos humanos en un área determinada. Este sistema pretende simular a un experto en música que compone dictados para que los estudiantes entrenen.

Los dictados que el sistema genere pueden ser de dos tipos, rítmicos y melódicos. Cada tipo de dictados posee una serie de niveles que confiere diferentes grados de dificultad a los dictados. Esto permite a los usuarios con menos conocimientos empezar por los niveles más elementales e ir avanzando según los conocimientos se consoliden.

Dependiendo del nivel y del tipo de dictado que el usuario decida que se genere, se trata de que el sistema genere un dictado en función de unos parámetros de configuración fijos y otros más o menos aleatorios. Esto permite que el sistema componga un dictado diferente cada vez, lo que proporciona al sistema mayor flexibilidad y un número amplio de posibilidades de resultados.

Un número alto de resultados variados en cada tipo y nivel permite al usuario entrenar tantas veces como sea necesario para dominar las características tratadas en esos dictados.

El sistema pretende ser sencillo y apto para cualquier usuario interesado en el entrenamiento del oído musical independientemente del nivel de conocimiento que tenga sobre el ámbito la informática y los sistemas expertos.

1.3. Estructura de la memoria

La memoria de este proyecto está estructurada en 6 capítulos.

El Capítulo 1 corresponde con la Introducción.

En el Capítulo 2 se presenta un conjunto de definiciones y conceptos musicales necesarios para el desarrollo y entendimiento del sistema.

En el Capítulo 3 se expone una visión general de la relación entre la inteligencia Artificial y la música así como un resumen de las tecnologías involucradas la creación de este sistema.

En el Capítulo 4 se describe el proceso de diseño del sistema. Se presentan los diferentes módulos que lo componen, la funcionalidad de los mismos y mediante qué lenguaje han sido desarrollados.

En el Capítulo 5 se describe la implementación del sistema. Se explica detalladamente las diferentes herramientas que se han utilizado en los diferentes módulos del sistema, se detallan las decisiones de diseño tomadas y el porqué de estas decisiones.

Se ha realizado una evaluación en dos fases, cuyos resultados exponen en el Capítulo 6. En la primera fase se ha comprobado el correcto funcionamiento del sistema. En la segunda fase, se ha comprobado la satisfacción de los usuarios.

Las conclusiones obtenidas y los trabajos futuros que podrían continuar este proyecto se describen en el Capítulo 6.

Finalmente, se presenta el Anexo A, en el que se muestran los resultados obtenidos con cada prueba realizada.

2. CONCEPTOS MUSICALES

En este capítulo se presentará una breve introducción a conceptos musicales generales de interés para la comprensión de este proyecto por parte de no expertos en música.

2.1. Notación musical

Notación musical [1] es el nombre genérico de cualquier sistema de escritura utilizado para representar gráficamente una pieza musical. El sistema de notación más utilizado actualmente es el sistema gráfico occidental que utiliza signos grabados sobre un pentagrama (conjunto de 5 líneas horizontales paralelas sobre las que se escriben las figuras). Existen muchos otros sistemas de notación.

El elemento básico de cualquier sistema de notación musical es la nota, que representa un único sonido y sus características básicas son duración y altura. Los sistemas de notación también permiten representar muchas otras características, tales como variaciones de intensidad, expresión o técnicas de ejecución instrumental.

El **pentagrama** [2] es el lugar donde se escriben las notas y demás signos musicales. Tiene cinco líneas y cuatro espacios, que se enumeran de abajo hacia arriba. Las líneas son equidistantes y horizontales.

Figura 1: Pentagrama

El pentagrama es una manera de realizar una notación musical de tal modo que la misma sea fácilmente transmisible a otras personas. Esto significa que los símbolos musicales se juntan en el pentagrama para formar una partitura que puede ser interpretada por un instrumento musical o cantada por la voz del ser humano. La **partitura** [3] es un sistema de notación musical. En la partitura cada instrumento y voz está escrito en uno

o varios pentagramas, y cuya simultaneidad en el tiempo se hace coincidir verticalmente.

El **compás** es una subdivisión del pentagrama que está delimitada en la partitura por líneas verticales y determinan la estructura rítmica de la música. La línea de tiempo se extiende de izquierda a derecha. En cada compás se podrán escribir un número determinado de una determinada figura. También se define como la unidad de tiempo que sirve para medir la duración de las notas musicales y de sus silencios.

A la izquierda del pentagrama de una partitura aparece un símbolo distintivo llamado **clave**. La función de esta clave es asociar las notas musicales con las líneas o espacios del pentagrama. La clave define qué nota ocupará cada línea o espacio en el pentagrama. Además, transpone toda la representación musical a una que se adecue mejor al instrumento que a irá a reproducir.

La figura siguiente es un símbolo que identifica a la "clave de Sol en segunda". Esto significa que la segunda línea del pentagrama empezando por abajo corresponderá a la nota Sol.

Figura 2: Clave de Sol en segunda al comienzo de un pentagrama.

Existe una variedad considerable de claves. Dependiendo de la amplitud del registro de notas que tenga cada instrumento, las partituras para cada uno de ellos utilizarán una clave u otra.

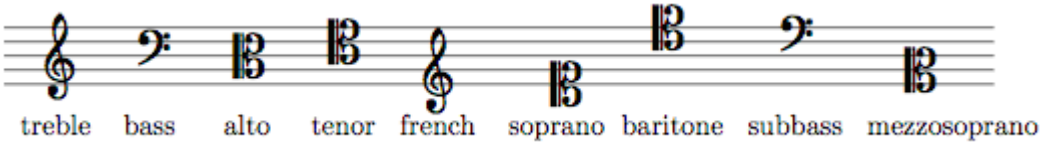


Figura 3: Conjunto de claves para distintos instrumentos o voces.

Las claves más utilizadas son la de Sol en Segunda, comentada más arriba, la clave de Fa en Cuarta y las claves de Do en tercera y cuarta.

Figura 4: Clave de Fa en cuarta.

Figura 5: Clave de Do en Tercera.

Justo después de la clave se escribirá el tiempo de compás del dictado.

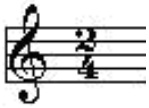


Figura 6. Pentagrama con Clave de Sol y tiempo del compás.

Figuras rítmicas o musicales: Las figuras rítmicas son símbolos que representan el tiempo de duración de las notas musicales y nos ayudan a subdividir el tiempo en la música. Sus símbolos son los mostrados en la figura siguiente:

Tabla 1. Figuras rítmicas.

Nombre	Símbolo
Redonda	
Blanca	
Negra	
Corchea	
Semicorchea	

Fusa	
Semifusa	

Cada figura vale el doble que la siguiente y la mitad de la anterior. Se puede ver mejor en la siguiente tabla y posterior esquema.

Tabla 2: Figuras musicales y duración de las mismas.

Nombre	Símbolo	Duración
Redonda		4 pulsos
Blanca		2 pulsos
Negra		1 pulso
Corchea		1/2 pulso
Semicorchea		1/4 pulso
Fusa		1/8 pulso
Semifusa		1/16 pulso

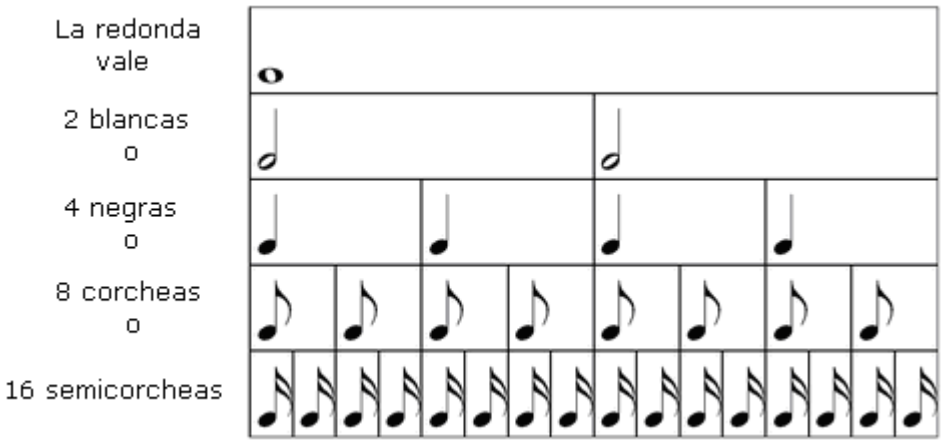


Figura 7: Esquema de correspondencia entre las figuras musicales.

Las **pausas** representan el *silencio*, esto es, el tiempo en que la voz o instrumento no produce sonido alguno, siendo llamados valores negativos. Las pausas se subdividen al igual que las notas en términos de duración. Cada pausa dura el mismo tiempo relativo que su nota correspondiente, o sea, la pausa más larga corresponde exactamente a la duración de una redonda. La correspondencia se hace en el siguiente orden:

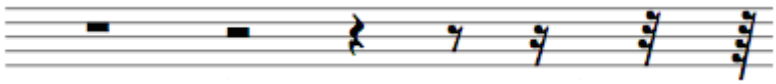


Figura 8: Símbolos de silencios en orden de duración de mayor a menor.

Tabla 3: Figuras musicales y sus silencios correspondientes.

Nombre	Símbolo	Silencio
Redonda		
Blanca		
Negra		
Corchea		
Semicorchea		
Fusa		
Semifusa		

2.2. El tiempo del compás

El tiempo del compás regula cuántas unidades de tiempo deben existir en cada compás. Se refiere a cuántas figuras de un determinado valor caben en cada compás.

Un compás se mide por medio de un quebrado con dos cifras o fracción de compás, que representa el tiempo de la partitura. La numerador

representa las unidades de tiempo que tiene el compás (en un 3/4, tres unidades o tres tiempos); el denominador, el valor de una nota en cada unidad o tiempo del compás. El número de la redonda es 1; el de la blanca 2; 4 el de la negra; 8 la corchea; 16 la semicorchea. O dicho de otra manera, el de la redonda es 1; el de la blanca 1/2; 1/4 el de la negra; 1/8 la corchea; 1/16 la semicorchea, ya que la duración en tiempo de cada figura es la mitad de la anterior.

Se pueden explicar de la siguiente manera:

$\left.\begin{matrix} X \\ Y \end{matrix}\right\}$ Indica que en el compás caben X figuras de Y valor

Por ejemplo, un compás en 3/4 tiene, por tanto, tres unidades de tiempo y cada una de ellas del valor de una negra. Como cada negra es equivalente a dos corcheas, también puede albergar seis corcheas, o dieciocho semicorcheas. Una negra ocupa el valor de cada unidad de tiempo del compás.

Los compases, según la cantidad de partes de las que constan, se pueden clasificar en binarios, ternarios o cuaternarios.

Los **compases binarios** se dividen en dos tiempos. A su vez, cada uno de estos pulsos se puede subdividir en dos o en tres corcheas, dando compases de subdivisión binaria o ternaria, respectivamente. Por ejemplo $\frac{2}{4}, \frac{6}{8}$, son tiempos binarios.

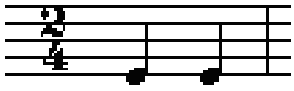


Figura 9: Ejemplo de compás binario con subdivisión binaria.

Figura 10: Ejemplo de compás binario con subdivisión ternaria.

En el ejemplo, se ve que cada compás se podría dividir en 2 tiempos. Y cada tiempo en dos o tres corcheas respectivamente.

Los **compases ternarios** se dividen en tres tiempos. A su vez, cada uno de estos pulsos se puede subdividir en dos o en tres corcheas, dando compases de subdivisión binaria o ternaria, respectivamente. Dentro de los tiempos ternarios están $\frac{3}{4}$, $\frac{9}{8}$.

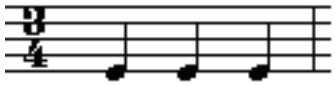


Figura 11: Ejemplo de compás ternario.

Los **compases cuaternarios** se dividen en cuatro tiempos. A su vez, cada uno de estos pulsos se puede subdividir en dos o en tres corcheas, dando compases de subdivisión binaria o ternaria, respectivamente. Por ejemplo $\frac{4}{4}$, $\frac{12}{8}$, son tiempos cuaternarios.

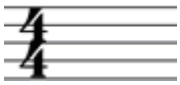


Figura 12: Tempo de "cuatro por cuatro". La unidad de tiempo tiene una duración de 1/4 y el compás tiene 4 unidades de tiempo.

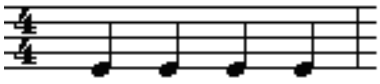


Figura 13: Ejemplo de compás cuaternario con subdivisión binaria.

2.3. Las notas

Las **notas** musicales son frecuencias de sonido que se han estandarizado. Únicamente existen siete nombres diferentes para denominar un amplio rango de frecuencias de notas. La correspondencia de nomenclaturas es la mostrada en la tabla siguiente:

Tabla 4: Correspondencia de las nomenclaturas de las frecuencias.

Nomenclatura	***Notas***
Anglosajona	**C D E F G A B**
Alemana	**C D E F G A H**
Española, italiana y francesa	**Do Re Mi Fa Sol La Si**

La manera de poder nombrar a todas las frecuencias mediante únicamente siete nombres, es repitiendo esa secuencia a lo largo de todas las frecuencias. Este tipo de secuencias de notas se denomina **escala**.

La altura de cada nota se representa por su posición en la pauta en referencia a la nota definida por la clave utilizada, como se muestra en la siguiente figura:

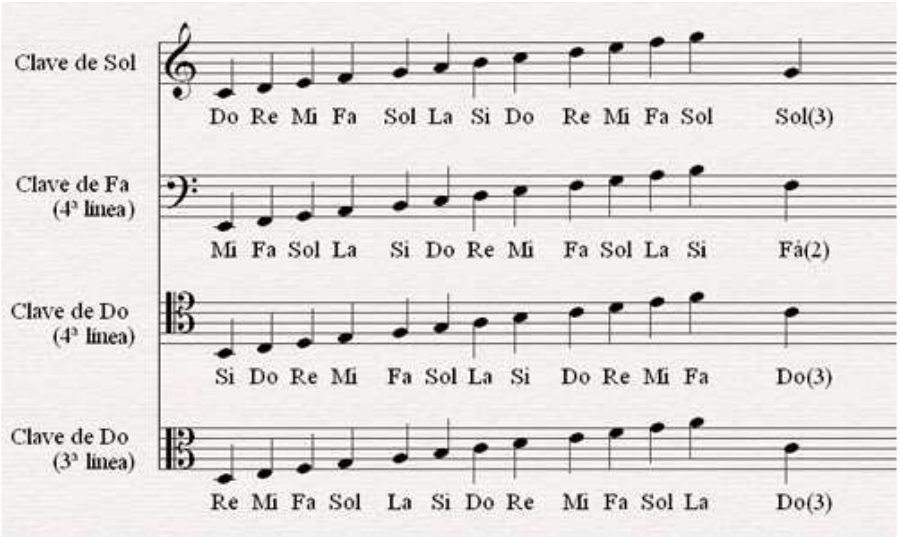


Figura 14: Correspondencia de las alturas entre las diferentes claves.

Existe una escala de referencia cuya relación notas-frecuencias es la mostrada en la tabla siguiente:

Tabla 5: Relación entre notas y frecuencias.

Do	Do#/Re*b*	Re	Re#/Mi*b*	Mi	Fa
261,63	277,18	293,66	311,13	329,63	349,23

Fa	Fa#/Sol*b*	Sol	Sol#	La	La#	Si
349,23	369,99	392,00	415,30	440,00	466,16	493,88

Esta escala es de referencia ya que es la que contiene la nota de referencia para cualquier instrumento musical, el *la 440*.

La 440 es el nombre que se le da coloquialmente al sonido que produce una vibración a 440 Hz y sirve como estándar de referencia para afinar la altura musical. El *la 440* es la nota musical la que se encuentra encima del *do* central del piano. En 1936, una conferencia internacional recomendó que el la que se encuentra a la derecha del do central del piano se afinara a 440 Hz. El estándar fue aceptado por la Organización Internacional de Estandarización en 1955 como ISO 16. Desde entonces ha servido como la frecuencia de sonido de referencia para la afinación de todos los instrumentos musicales (pianos, violines, etc.) y se utiliza prácticamente en todo el mundo.

El estándar inicial era *la 439* Hz, pero fue reemplazado por el *la 440* Hz después de registrarse quejas acerca de la dificultad de reproducir los 439 Hz en los laboratorios debido a que 439 es un número primo.

Una manera de diferenciar unas frecuencias de otras denominadas con el mismo nombre, es añadiendo un sufijo numérico al nombre de la nota. Así, por ejemplo, el Do escala anterior es también denominado C4.

Además, existen las **dislocaciones de tono**, accidentes o alteraciones musicales. Estos son el sostenido (#), el bemol (*b*), el doble sostenido (x) y el doble bemol (*bb*). Se representan siempre *antes del símbolo* de la nota cuya altura será modificada y *después del nombre* de las notas, cifras y tonalidades. Un sostenido disloca o aumenta la nota medio tono arriba, un doble sostenido disloca el sonido un tono arriba, un bemol disloca la nota medio tono abajo y el doble bemol disloca el sonido un tono abajo. Por ejemplo, se puede decir que un "*Fa sostenido*" (Fa#) es la misma nota que un "S*ol bemol*" (Sol *b*) en el piano.

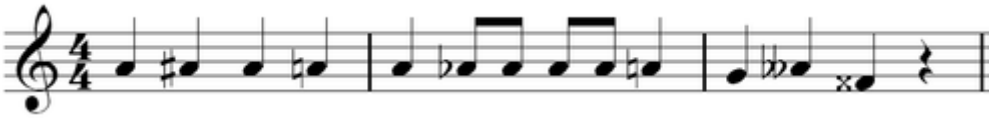


Figura 15: Conjunto de notas con alteraciones musicales.

Una vez que un sostenido o bemol haya sido aplicado a una nota, todas las notas de la misma altura mantendrán la alteración hasta el fin del compás. En el compás siguiente, todos los accidentes pierden su efecto, por lo que en caso de necesitarlo habrá que colocarlo nuevamente. Si se desea anular el efecto de un accidente aplicado inmediatamente antes de una clave de tonalidad, se debe usar un **becuadro** (♮) que devuelve a la nota su tono natural. En el ejemplo visto arriba se puede notar que la tercera nota del primer compás también es sostenida, pues el accidente aplicado a la nota anterior permanece válido y sólo es anulado por el becuadro que hace que la cuarta nota sea nuevamente un La normal. En el tercer compás, se puede ver un sol, un la doble bemol y un fa doble sostenido. Aunque tengan nombre diferente y ocupen distintas posiciones en la clave, los accidentes aplicados hacen que las tres notas suenen exactamente iguales.

2.4. Escalas

Las notas se agrupan de doce en doce, de las cuales siete son naturales, las mencionadas anteriormente, y cinco alteradas con sostenidos o bemoles.

Estos grupos de doce notas se repiten de izquierda a derecha (desde la nota más grave a la más aguda). Cada ocho notas naturales se cierra un grupo y se abre otro, y la distancia musical entre esas teclas se llama **octava** (se llama octava también el mismo grupo de 12 teclas), y su escala es igual a 2:1. Esto es, la frecuencia de la misma nota de la siguiente octava es el doble, y la de octava anterior es la mitad. La distancia de dos octavas le corresponde a la relación de frecuencias de 4:1, tres octavas 8:1 etc. Para sumar distancias se tiene que multiplicar las relaciones de frecuencias.

Esta ordenación de los sonidos musicales ha sido fruto de un largo proceso. Desde la elección de un sonido base, a partir del cual construir el resto, a la determinación del intervalo que hay entre una nota y la siguiente. En la figura siguiente se observan los intervalos entre las diferentes notas.

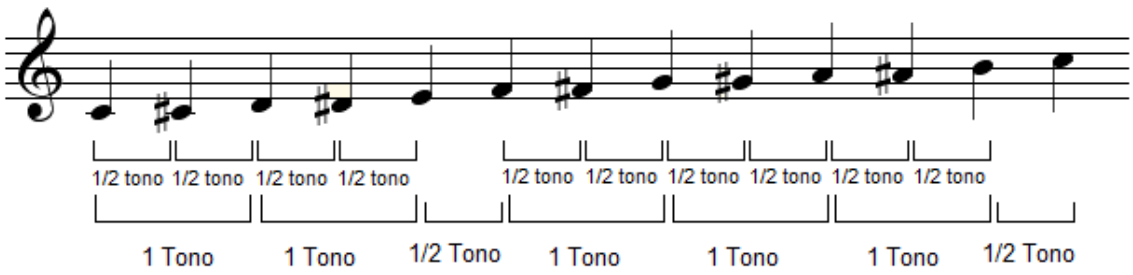


Figura 16: Distancias entre notas de la escala de Do M.

Como se puede ver en la figura, la distancia entre todas las notas con diferente nombre es de un tono, excepto entre los pares de notas Mi-Fa y Si-Do, entre los que hay únicamente medio tono.

Entre una nota y su alteración superior (sostenido) o inferior (bemol), la distancia es de medio tono.

Así, una **escala** es una sucesión de sonidos constitutivos de un sistema que se suceden regularmente en sentido ascendente o descendente, y todos ellos con relación a un solo tono que da nombre a toda la escala. Ese tono se denomina tónica.

Se puede definir, además, la **tonalidad** como la organización musical que se define por el orden de los intervalos dentro de la escala de los sonidos. Además, cada tonalidad indica las distintas alteraciones fijas, sostenidos y bemoles, que va a tener la escala.

Estas alteraciones se indican al principio de la partitura. Las tonalidades puedes ser Mayores, menores, de séptima, etc. Estas dependen de la secuencia de tonos (T) y semitonos (S). Una escala mayor deberá tener la secuencia: T-T-S-T-T-T-S, mientras que una escala menor deberá tener la secuencia: T-S-T-T-S-T-T.

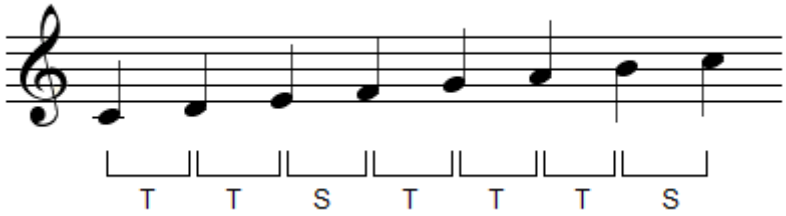


Figura 17: Escala de Do Mayor. Sin alteraciones.

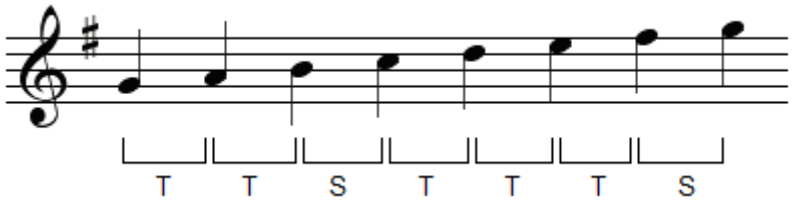


Figura 18: Escala de Sol Mayor con su única alteración fija, Fa.

Como se puede observar, aunque ambas escalas empiezan en diferente nota, la secuencia de tonos y semitonos es idéntica en ambos casos, ya que ambas escalas son mayores.

La siguiente imagen muestra una escala ascendente y descendente de la tonalidad de Do M, también llamada escala diatónica. Las ocho primeras notas hacen referencia a una octava, como se ha explicado anteriormente.

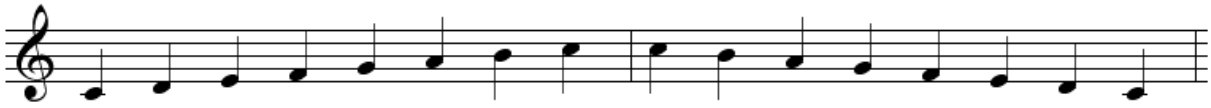


Figura 19: Escala diatónica.

2.5. Dinámica

En música, la dinámica se refiere a las gradaciones de la intensidad de la música. Existen al menos ocho indicaciones de **dinámica**, empezando desde un sonido muy quedo, hasta un sonido muy fuerte.

Intensidad	**Nomenclatura**	**Nombre italiano**
Más débil ↓ Más fuerte	*ppp*	Pianíssimo piano o pianississimo
	pp	Pianissimo
	p	Piano
	mp	Mezzo piano
	mf	Mezzo forte
	f	Forte
	ff	Fortísimo
	fff	Fortíssimo forte o fortississimo
	Silencio	

Si se reduce la elección a dos únicas posibilidades, la nota sonaría fuerte o débil. Correspondería a *forte* (mayor intensidad) o *piano* (menor intensidad).

La ejecución de los matices es totalmente subjetiva y, por tanto, dependen del músico y del estilo o periodo histórico correspondiente a la obra, además de la forma de consideración personal ante esta dinámica. No hay ninguna forma racional de medir la dinámica del sonido. Además, de debe notar que las indicaciones de dinámica son relativas, no absolutas. Por ejemplo p (*piano*) no indica un nivel exacto de intensidad de la música, sino que un cierto pasaje debe ser un poco más suave que f (*forte*).

Sin embargo, en los dictados, al ser una herramienta de aprendizaje, la dinámica no es tanto un matiz subjetivo como un instrumento para poder diferenciar entre los distintos tiempos anteriormente explicados.

De esta manera, en los compases binarios, el primer tiempo se reproducirá con dinámica *forte* y el segundo *piano.*

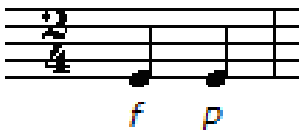


Figura 20: Dinámica de un compás binario.

En los tiempos ternarios, el primero será *forte* y el segundo y tercero, *piano.*

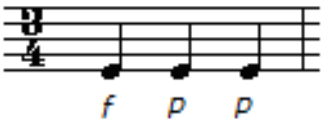


Figura 21: Dinámica de un compás ternario.

En los tiempos cuaternarios, se tiene la sucesión *forte*, *piano*, *mezzoforte, piano*. Aunque este último caso, normalmente se simplifica asemejándolo a los anteriores, el primero *forte,* y el resto *piano*.

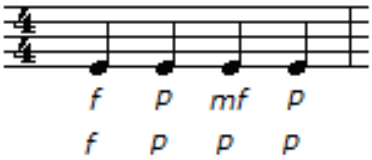


Figura 22: Dinámica de un compás cuaternario.

3. ESTADO DEL ARTE

3.1. La inteligencia artificial y la música

3.1.1. Introducción

Según la definición clásica, la Inteligencia Artificial (IA) es el **estudio de las facultades mentales a través del uso de modelos computacionales** [3]. Una tarea fundamental en la IA consiste en expresar el conocimiento de la realidad a través de un lenguaje conveniente de tal manera que pueda ser manipulada por una máquina inteligente y producir nuevos conocimientos de la realidad. Un campo particular de la IA es el denominado Representación del Conocimiento (RC), cuyos objetivos son el desarrollo de modelos adecuados para la captura y manipulación del conocimiento.

Dentro de la IA hay diferentes teorías contrapuestas; por ejemplo, una teoría "lógica" expone que las manipulaciones deberían corresponder a mecanismos de inferencia en sistemas de lógica formal (representación del conocimiento de la realidad). En el lado contrario, la teoría "conexionista", más reciente que la anterior, afirma que las "máquinas inteligentes" deberían reflejar la estructura del cerebro y en las cuales el "conocimiento" no estaría expresado explícitamente en ningún tipo de lenguaje sino que sería "aprendido" por la máquina como estados-conexiones particulares entre unos elementos de procesamiento (neuronas) relativamente simples.

El término genérico de Sistemas de Inteligencia Artificial (SIA) se refiere a las implementaciones en ordenadores de modelos que se basan en hipótesis conexionistas tales como las descritas anteriormente. No obstante un programa informático basado en IA podría no ser un SIA; por ejemplo, programas informáticos desarrollados usando lenguajes típicos de programación en IA no se deberían considerar como "inteligentes" tan sólo porque utilizan este tipo de lenguajes. Al contrario, determinados programas informáticos que usan tipos de lenguaje procedimentales, tales como PASCAL o C, podrían ser SIA. De la misma forma, un sistema basado en técnicas de programación orientadas a objetos (POO) no es un SIA pues

no tiene características "inteligentes", como por ejemplo la capacidad de inferencia. Aunque la POO ha demostrado que es una buena plataforma para el desarrollo de SIA.

Otro conocido término de IA es el de "sistema basado en el conocimiento". Un SIA no se convierte en un sistema basado en conocimiento tan sólo porque exista algo en el sistema que toma el conocimiento y puede escribirlo o expresarlo; tampoco lo es por el hecho de que se comporte como si tuviera inteligencia; se convierte en un "sistema basado en el conocimiento" cuando la arquitectura contiene bases de conocimiento explícito. Para ello, el sistema debería contar con algún lenguaje bien especificado para codificar sus adquisiciones. Este papel es jugado por un lenguaje de representación del conocimiento. Desde este enfoque se pueden identificar los tres elementos básicos para elaborar un sistema de representación del conocimiento:

- el lenguaje de representación
- el mecanismo de inferencia
- un dominio específico de conocimiento

Por otra parte, la música está normalmente relacionada con la expresión de emociones, algo puramente humano, jugando el intelecto humano un importante papel en actividades musicales.

Se puede definir la música como una actividad intelectual, como una habilidad esencial de la mente humana que requiere sofisticados mecanismos de memoria involucrando tanto manipulaciones conscientes de conceptos como accesos subconscientes a millones de neuronas pertenecientes a una red compleja. Así, se considera que las reacciones emocionales a la música provendrán de algún tipo de actividad intelectual.

Pero la música no es percibida exclusivamente a través de los oídos, sino también a través de otros sentidos. La actividad cerebral en respuesta a los estímulos puede medirse mediante la actividad de las neuronas, encontrando así dos métodos comunes de medida:

- **PET (Positron Emission Tomography)**: mide la actividad cerebral escaneando el flujo de un material radiactivo previamente insertado en el flujo sanguíneo del individuo observado. A pesar de su eficiencia no es un método muy popular por el desconocimiento de los posibles efectos secundarios que dicho material pueda tener en la salud. Proporciona una sección clara del área del cerebro donde el flujo sanguíneo es más intenso durante el proceso auditivo.

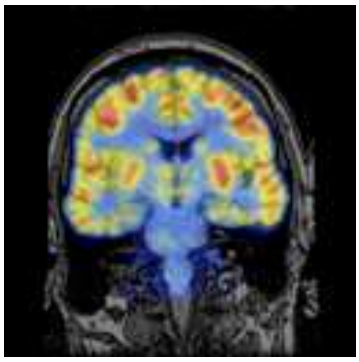


Figura 23: PET.

- **ERP (Event Related Potential):** utiliza pequeños electrodos en contacto con el cráneo para medir la actividad eléctrica del cerebro, es más seguro que PET. Da un nivel de voltaje en el tiempo de la actividad eléctrica de las áreas del cerebro donde los electrodos han sido colocados.

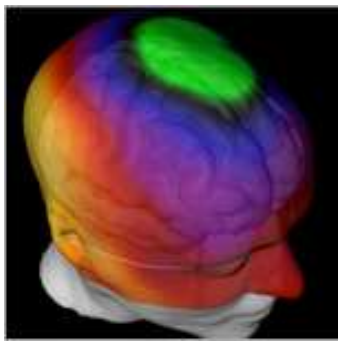


Figura 24: ERP.

Estas pruebas físicas se llevan a cabo para comprender mejor la inteligencia y su metodología, uno de los objetivos de la IA, cuya

metodología se aleja de este tipo de pruebas físicas para basarse en la lógica, en modelos matemáticos y en simulaciones informáticas de comportamientos inteligentes.

Las matemáticas y la lógica, inherentemente ligadas a procesos musicales, juegan un papel fundamental en la formalización de lo que se considera inteligencia. La gran mayoría del trabajo de IA desarrollado hasta el momento asume que la inteligencia puede simularse encapsulando conjuntos de datos en paquetes estáticos de información. La actividad inteligente se lleva a cabo mediante un mecanismo que coge y combina paquetes apropiados de información almacenada en memoria para llegar a objetivos específicos. En este sentido el conocimiento se clasifica en dos grupos:

a) **Declarativo:** por ejemplo la semántica de la gramática o el significado de los paquetes de información.
b) **Procedimental:** por ejemplo la gramática en sí misma o cómo ese mecanismo de tratamiento de la información debería funcionar.

El principal objetivo de las primeras investigaciones en IA fue reducir los procesos de razonamiento matemático a procesos de lógica formal. Las aproximaciones lógicas aplican los mecanismos de inferencia lógica a problemas de representación y razonamiento de la realidad. El problema es que esta posición asume que los formalismos lógicos tienen éxito en dominios formales, tal como el de las matemáticas. Pero se arguye que las teorías simbólicas -lógicas y analógicas- carecen de plausibilidad cognitiva debido a que sólo existen acercamientos toscos a las actividades del cerebro.

Otra posición ha sido bautizada como "*conexionismo eliminativo*", dado que elimina el nivel simbólico, y es que diversos formalismos lógicos han sido explícitamente desarrollados o aplicados a la música.

Existen asimismo ciertas dificultades en el uso de la lógica formal en la representación del conocimiento que han dado lugar a los sistemas

basados en subconjuntos híbridos de lógica de primer orden. Se ha tratado de conseguir un equilibrio entre expresividad y tratabilidad computacional (la dificultad de razonar correctamente con un lenguaje representacional aumenta a medida que aumenta el poder expresivo del lenguaje).

La IA y la música se encuentran por así decirlo en extremos opuestos del espectro, por un lado la IA es la personificación de la ciencia computacional mientras que la música es la personificación del arte y la abstracción.

La creación de música dinámica y autónomamente tiene infinitas posibilidades, piezas que pueden ser compuestas en segundos, brillantes acompañamientos para guitarristas, músicos de jazz, etc., que ayudan a desarrollar su estilo.

La primera inclinación para usar casi cualquier tecnología, es duplicar funciones ya existentes para demostrar la utilidad en dicha tecnología o para eliminar los métodos tradicionales de llevar a cabo la función (cuando estos son tediosos, peligrosos o indeseables). El segundo camino de utilización de la tecnología es para llevar a cabo funciones deseables pero previamente irrealizables, y por último una tercera motivación para el uso de una tecnología es descubrir nuevas funciones desconocidas previamente.

Una gran parte del trabajo realizado con música en relación con los ordenadores implica el primer modo de utilización comentado tratando de hacer que los ordenadores se comporten como simuladores humanos.

3.1.2. Áreas de trabajo

A la hora de hacer música, las únicas actividades que pueden considerarse realmente tediosas son la práctica instrumental técnica y la copia de música, aunque por supuesto existen más campos dentro de la musicología en los que la influencia de los ordenadores ha permitido avanzar en ciertos sentidos.

Tabla 6. Aplicaciones de los ordenadores a la música.

Generación de música de manera automática
Aumentar la velocidad a la hora de crear música
Interpretar o ensayar
Improvisar
Transcripción de música automática

Se han llevado a cabo numerosos intentos de reducir la pesada tarea de imitar música, mejorando la velocidad a la hora de hacerlo y su calidad. También se ha intentado pero con menos éxito duplicar otros aspectos de la generación de música humana como componer, ensayar, interpretar, improvisar, escuchar...

Muchas de estas acciones el ser humano las realiza con gusto, por lo que duplicarlas con un ordenador podría corresponder,

a) a un **enfoque técnico** pero sin ninguna intuición creativa como guía.
b) También podría corresponderse por un **enfoque científico básico.** Pero los beneficios específicos de esta aproximación en el relativamente joven y especializado campo de la música por ordenador son pocos.
c) Una tercera aproximación, particularmente aplicable a la investigación en inteligencia musical artificial es aquella llamada **psicología aplicada**. Su interés es el uso de los ordenadores como una herramienta para programar y explorar modelos de conocimiento de inteligencia humanas. Mantiene que las teorías acerca de la inteligencia humana pueden ser modeladas mediante programas informáticos o que los modelos de programación pueden permitir mirar dentro de los procesos intelectuales y cognitivos.

Los compositores han utilizado los ordenadores para llevar a cabo su concepción de músicas irrealizables por los humanos o como una herramienta para desarrollar ideas de composición que requerían cantidades de cálculos impensables sin la ayuda de ordenadores. Definiendo y programando nuevas funciones se pueden mejorar operaciones de

composición e instrumentales, incrementando así el número de habilidades a desarrollar por el músico.

Las exploraciones en el campo de la música informatizada se pueden dividir en dos grandes categorías: de entrada y de salida. Estas dos categorías se pueden hacer corresponder con dos comportamientos intelectuales musicales: conocimiento y composición musicales.

3.1.3. Generación de música de forma computerizada

La creación de música requiere cierto grado de inspiración, pero es muy difícil imprimir esa inspiración a los ordenadores. Rara vez se ha visto un programa generador de música que no tome como entrada algo proporcionado por los humanos. La mayoría de los programas actuales requieren que cierto usuario humano establezca ciertos parámetros y opciones que son utilizadas por la máquina para generar la pieza musical. Una gran cantidad de programas de música utilizan algoritmos genéticos como medios de generación de estas piezas.

Otros programas utilizan varios módulos que se comunican unos con otros, algunos controlados por algoritmos genéticos y otros suelen utilizar fórmulas matemáticas para generar la pieza musical.

Los principales problemas que aparecen a la hora de componer música artificialmente pasan por determinar qué se considera como buena música o generar una serie de sonidos que tengan sentido musical a la hora de juntarlos. Para el primero de estos problemas habrá muchos criterios por lo que la programación de este tipo de decisiones se hará difícil, mientras que para el segundo existe lo que se llama música fractal.

3.1.4. Transcripción automatizada de la música

El cerebro humano tiene la capacidad de, a través del oído, escuchar un cierto sonido ignorando (o no prestando demasiada atención) a otros que estén superpuestos a él.

Desarrollar un programa que sea capaz de afinar para separar los distintos sonidos implicados es implícitamente difícil, ya que se desconoce

como el cerebro es capaz de distinguir unos sonidos entre otros. En este campo el reconocimiento de voz será la que algún día nos dé algunas respuestas, ya que el reconocimiento de voz es básicamente el estudio de encontrar significado dentro de un sonido, mientras que la auto transcripción sería encontrar significado o distinción en una multitud de sonidos.

3.1.5. Tiempo y lógica

En la música, el tiempo tiene un papel fundamental. El **razonamiento temporal** (RT), ha sido muy útil en la constatación del papel del tiempo en la representación del conocimiento y el razonamiento. Las teorías musicales deberían contrastarse con las teorías formales del tiempo, y en este sentido el RT adquiere gran importancia.

La importancia del RT se muestra a través de determinadas tareas que requieren una explícita representación y razonamiento temporal:

a) **predecir** determinados estados del mundo.
b) **explicar** cómo ha llegado el mundo a su presente.
c) **encontrar** una serie de acciones a ejecutar para alcanzar una situación deseada en el mundo.
d) **encontrar** las reglas que determinan cambios en el mundo.

Todas estas tareas son de crucial importancia en la representación musical. Por ejemplo, un estudio descubrió que el entrenamiento musical usando el teclado aumenta a largo plazo el razonamiento temporal. Existe, además, el llamado *Efecto Mozart* [5]. Éste se refiere a los efectos que puede producir a personas escuchar las melodías de W. A. Mozart. Según algunas investigaciones, escuchar música de Mozart se traduce en una pequeña elevación de la puntuación de ciertos tests que evalúan el razonamiento temporal de quien la escucha. Esto es debido a las ppm (pulsaciones por minuto) que tiene en especial la música de Mozart, ya que cambian el estado del cerebro y lo hacen más receptivo.

3.1.6. Inteligencia Artificial y conocimiento musical

Los intentos para modelar el conocimiento musical con IA se aproximan normalmente a un incremento del conocimiento de la psicología humana y el intelecto.

El conocimiento informático de la música implica cuatro problemas:

a) Cómo se mide la música para proporcionar información de entrada al sistema informático
b) Cómo se presenta esa información al ordenador
c) Cómo esa información será representada dentro del programa informático para que dicho programa pueda llegar a algún entendimiento de su significado
d) Qué hará el ordenador con este conocimiento

En la teoría musical de la música Occidental lo importante se debe entender como un conjunto de dimensiones paramétricas simultáneas separadas: tono, duración, instrumentos..., cuyo método de medida tiene una gran dependencia de la cultura. No obstante, la aproximación paramétrica y las unidades de medida son en muchos casos grandes generalizaciones.

No se puede tener control de todos los parámetros existentes en una pieza musical, por lo que se debe decidir uno o más parámetros a medir, pero teniendo en cuenta que la manera de definirlos y escogerlos está basada en la cultura, estilo musical e incluso preferencias personales del autor o el oyente.

Una vez que se ha decidido lo que se va a medir dentro de una pieza musical, la siguiente decisión trata acerca de cómo medir esos aspectos. Además, se debe decidir el grado de precisión al que se quiere llegar.

La decisión dependerá probablemente de si se quiere proporcionar al sistema de la máxima cantidad posible de información o si por el contrario es suficiente con un mínimo aceptable. Claramente, a medida que la entrada de información aumenta, así lo hará también el potencial de detalle

en la representación. Dependiendo de lo que se quiera medir será bueno un nivel de detalle u otro.

Por ejemplo, en una medida del tono (pitch) y la duración de un fragmento musical, si la intención fuese reproducir el sonido original, sería deseable la máxima cantidad de información, mientras que si lo que se quiere es reproducir la notación, un menor nivel de precisión sería más adecuado.

En general, es deseable que la medida de entrada tenga la mayor cantidad de detalle permitida por el sistema de representación, asumiendo que el programa deducirá la información fundamental de manera algorítmica.

Sin embargo, de manera más práctica, uno de los caminos más comunes en que los diseñadores de modelos cognitivos miden música es utilizando un controlador MIDI para capturar los datos midiendo las "formas" del sonido.

Para examinar las formas de manejar datos obtenidos a partir de fragmentos musicales, se considera un problema concreto de cognición/conocimiento como es la percepción del ritmo.

Percepción del Ritmo

Los intervalos entre teclas pulsadas se representan inicialmente como una cadena de números que muestran la medida en algo absoluto como pueden ser milisegundos. Esta cadena de números es lo que se llama **ritmo**, pero para que tenga algún significado musical deben detectarse patrones en estas cadenas y el método que a priori parece más obvio para los músicos es tratar de comparar esta cadena de valores con ritmos conocidos de notación parecida. En caso de encontrar un ritmo parecido, los números de lo que se ha detectado pueden ajustarse hacia ese ritmo conocido y expresarse en términos relativos en lugar de absolutos.

Tabla 7: Valores relativos de las figuras musicales.

Nombre de la figura rítmica	Valor numérico
Redonda	4.0
Blanca	2.0
Blanca con puntillo	3.0
Negra	1.0
Negra con puntillo	1.5
Corchea	0.5
Corchea con puntillo	0.75
Semicorchea	0.25
Semicorchea con puntillo	0.375

El ritmo es percibido detectando patrones de eventos en el tiempo. Cualquier método de detección de estos patrones puede emplearse para procesar los datos musicales, para agruparlos en eventos musicales destinados a estar en el mismo patrón.

Los intervalos de tiempo dentro de los cuales se encuentran los patrones de este tipo de eventos se usan para lanzar hipótesis acerca de cuál será el ritmo percibido en una pieza de música. Entonces dicho ritmo es analizado para obtener conceptos organizacionales tales como el pulso o la métrica.

El patrón más básico para analizar es la detección aislada un evento cualquiera. Si por evento se considera la presión de una nota, se podrá realizar hipótesis sobre un ritmo basándose en cadenas de intervalos de tiempo entre pulsaciones de notas. Al ser el más simple y obvio de los indicadores de ritmo es también el más usado.

Para determinar un ritmo con el que comparar, el oyente tiene que determinar primero un intervalo básico de tiempo que debe permanecer constante durante un cierto periodo. Se generará entonces el intervalo entre notas pulsadas para ajustarse a algún múltiplo entero o división del intervalo básico.

En la mayoría de las implementaciones informáticas, la unidad de medida es considerablemente menor que el intervalo más pequeño que puede ser considerado un pulso musical. Por lo tanto los datos de entrada han de ser cuantificados. La mayoría de los secuenciadores MIDI utilizan un método simple de redondeo.

Las discrepancias que existan entre el ritmo trabajado y el ritmo teórico serán debidas a tres factores. Los dos primeros son errores que forman parte de las desviaciones no intencionadas respecto a los valores teóricos.

- El primero, son las **pequeñas desviaciones** que ocurren inevitablemente debidas al error motor, un tipo de ruido matemático que puede considerarse insignificante.
- El segundo, es el denominado **error conceptual**. Cuándo tocar una nota está basado en una estimación del momento adecuado y en este proceso de estimación pueden ocurrir y ocurren errores.
- El tercer factor es el **error intencionado**, también conocido como rubato o temporización expresiva, interacción entre la temporización y otros parámetros expresivos.

Estas últimas desviaciones pueden ser bastante más amplias que las anteriores, dependiendo del estilo musical.

Casi todos los detectores de ritmo trabajan sobre la idea de "expectativa". Basándose en los ritmos percibidos hasta el momento se hacen predicciones de los momentos temporales en el futuro en los que es probable que ocurran nuevos eventos rítmicos, siendo estas hipótesis reforzadas o debilitadas a medida que el ritmo va coincidiendo o no con tales predicciones. Estas alteraciones con respecto a las hipótesis harán que éstas vayan modificándose en favor de reconocer cada vez mejor los ritmos y melodías analizados.

Al margen de los intervalos entre pulsaciones, hay muchos más factores que determinan la percepción del ritmo, y que están disponibles para ser incluidos en un algoritmo de detección del ritmo.

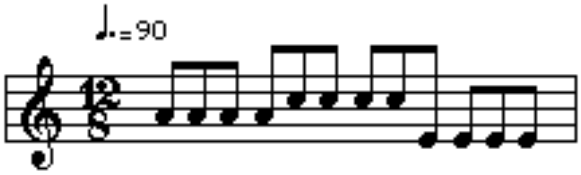


Figura 25: Partitura con diferentes análisis.

Un detector de ritmo que evaluase solamente los intervalos entre pulsaciones derivaría en una figura con doce notas constantes a partir de esta partitura. Sin embargo en esta pieza, el verdadero interés del ritmo está en los acentos dinámicos y la envolvente del tono que presenta dos análisis diferentes, que hay un acento dinámico cada tres octavos de nota y un cambio en el tono cada cuatro octavos de nota.

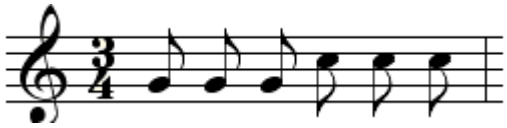


Figura 26: Partitura con diferentes análisis.

Si el análisis determinara únicamente los intervalos entre pulsaciones, de la figura anterior se podría derivar una única figura de seis notas constantes, tal y como muestra la figura siguiente.

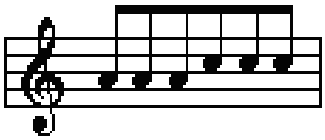


Figura 27: Análisis de intervalos.

Si se tienen en cuenta los acentos dinámicos, se puede observar un acento dinámico cada dos corcheas o notas.

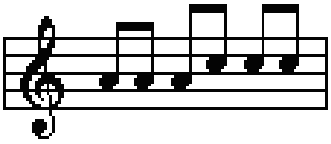


Figura 28: Análisis de los acentos dinámicos.

Si el análisis se centra en la envolvente del tono, se observaría un cambio de tono cada tres notas, como se muestra en la siguiente figura.

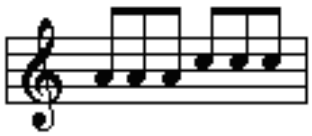


Figura 29: Análisis de la envolvente.

Ningún sistema conexionista admite las síncopas o desplazamientos del acento musical a notas normalmente no acentuadas como una posibilidad rítmica válida, lo cual es bastante limitador a la hora de tener en cuenta ciertos estilos musicales que utilizan ritmos sincopados que los caracterizan, sin indicación alguna del pulso.

Se puede concluir que un algoritmo que no tiene en cuenta el conocimiento del estilo es más general y objetivo.

3.1.7. Inteligencia Artificial y composición musical

Normalmente, se ha estudiado y documentado más los aspectos de cómo se realizan las actividades de composición musical y menos aquellos aspectos que tratan de por qué se hace o qué posibilidades existen en este campo.

La idea de que las decisiones pueden ser explicadas de manera algorítmica es el corazón de la IA, dado que los ordenadores sólo saben de cómo se hacen las cosas.

Hay que distinguir entre decisiones estéticas y otro tipo de decisiones. Las decisiones estéticas son aquellas encaminadas a la estética bajo criterios estéticos. Cuando el objetivo es la estética, se busca que el resultado sea agradable, concepto bastante ambiguo.

A la hora de componer, el compositor utilizará varios criterios estéticos aunque puede responder a sistemas preestablecidos. A la hora de componer música con el ordenador, el compositor seguiría reglas establecidas de toma de decisiones, algo que el ordenador haría mejor y

más rápido que los humanos. Aun así, la existencia de esas reglas implica algunas decisiones estéticas.

Sólo se puede decir que la música ha sido compuesta por un programa informático si dicho programa toma realmente decisiones. Estas decisiones pueden ser de carácter aleatorio, basadas en bases de conocimiento de valores estéticos, o basadas en conocimiento adquirido. Si no se incluyen elementos de decisión y simplemente el programa ejecuta una serie de reglas definidas, lo único que se demuestra es que el ordenador es una mejor máquina de cálculo que el ser humano, no que se esté comportando de manera inteligente.

El problema a la hora de intentar escribir un algoritmo de composición de uso general reside en que hay al menos tantas formas de hacerlo como compositores, y la mayoría de los compositores no estarán por la labor de usar un algoritmo desarrollado por otra persona. Esto en principio significa que un compositor con ideas acerca de cómo utilizar un ordenador para componer debe aprender a programar o colaborar con alguien que haga la tarea de programar los algoritmos específicos. Es muy difícil ser experto tanto en composición musical como en programación informática, por lo que la colaboración entre músicos y programadores parece ser un buen camino para hacer música artificial.

3.2. Sistemas Expertos

Un **Sistema Experto (SE)** [6] puede definirse como un sistema informático (hardware y software) que simula a los expertos humanos en un área de especialización dada.

Como tal, un sistema experto debería ser capaz de procesar y memorizar información, aprender y razonar en situaciones deterministas e inciertas, comunicar con los hombres y/u otros sistemas expertos, tomar decisiones apropiadas, y explicar por qué se han tomado tales decisiones. Se puede pensar también en un sistema experto como un *consultor* que puede suministrar ayuda a (o en algunos casos sustituir completamente) los expertos humanos con un grado razonable de fiabilidad.

Con los sistemas expertos se busca una mejor calidad y rapidez en las respuestas dando así lugar a una mejora de la productividad del experto.

Un SE puede entenderse como una rama de la IA. Estos sistemas imitan las actividades de un humano para resolver problemas de distinta índole, no necesariamente de inteligencia artificial. También se dice que un SE se basa en el conocimiento declarativo (hechos sobre objetos, situaciones) y el conocimiento de control (información sobre el seguimiento de una acción).

Para que un sistema experto sea herramienta efectiva, los usuarios deben interactuar de una forma fácil, reuniendo dos capacidades para poder cumplirlo:

a) **Explicar sus razonamientos o base del conocimiento**: los sistemas expertos se deben realizar siguiendo ciertas reglas o pasos comprensibles de manera que se pueda generar la explicación para cada una de estas reglas, que a la vez se basan en hechos.
b) **Adquisición de nuevos conocimientos o integrador del sistema**: son mecanismos de razonamiento que sirven para modificar los conocimientos anteriores. Sobre la base de lo anterior se puede decir que los sistemas expertos son el producto de investigaciones en el campo de la inteligencia artificial ya que esta no intenta sustituir a los expertos humanos, sino que se desea ayudarlos a realizar con más rapidez y eficacia todas las tareas que realiza.

Principalmente existen tres tipos de sistemas expertos:

- **Basados en reglas**: Los sistemas basados en reglas trabajan mediante la aplicación de reglas, comparación de resultados y aplicación de las nuevas reglas basadas en situación modificada.
- **Basados en casos o CBR** (Case Based Reasoning): El Razonamiento basado en casos es el proceso de solucionar nuevos problemas basándose en las soluciones de problemas anteriores.

- **Basados en redes bayesianas**: Una red bayesiana, o red de creencia, es un modelo probabilístico multivariado que relaciona un conjunto de variables aleatorias mediante un grafo dirigido que indica explícitamente influencia causal.

Un Sistema Experto está conformado por:

- **Base de conocimientos** (BC): Contiene conocimiento modelado extraído del diálogo con el experto.
- **Base de hechos** (Memoria de trabajo): contiene los hechos sobre un problema que se ha descubierto durante el análisis.
- **Motor de inferencia**: Modela el proceso de razonamiento humano.
- **Módulos de justificación**: Explica el razonamiento utilizado por el sistema para llegar a una determinada conclusión.
- **Interfaz de usuario**: es la interacción entre el SE y el usuario, y se realiza mediante el lenguaje natural.

3.2.1. Importantes SE

- **Dendral**: Dendral [7] es un sistema experto desarrollado por Edward Feigenbaum en la Universidad de Stanford, a mediados de los años 60, y su desarrollo duró diez años, (1965 a 1975). Fue el primer sistema experto en ser utilizado para propósitos reales, y el sistema tuvo cierto éxito entre químicos y biólogos, ya que facilitaba enormemente la inferencia de estructuras moleculares, dominio en el que Dendral estaba especializado.

- **Mycin**: Mycin [7] es un sistema experto desarrollado a principios de los años 70 por Edgar ShortLiffe, en la Universidad de Stanford. Fue escrito en Lisp, e inicialmente estaba inspirado en Dendral, Su principal función consistía en el diagnóstico de enfermedades infecciosas de la sangre; además, Mycin era capaz de "razonar" el proceso seguido para llegar a estos diagnósticos, y de recetar medicaciones personalizadas a cada paciente.

- **XCon**: XCon [7] era un sistema de producción basado en reglas escrito en OPS5 por John P. McDermott de CMU en 1978 para asistir a los

pedidos de los sistemas de computadores VAX de DEC (Digital Equipment Corporation) seleccionando los componentes del sistema de acuerdo a los requerimientos del cliente.

- **Dipmeter Advisor**: Dipmeter Advisor [8] fue un sistema experto temprano desarrollado en 1980 por Schlumberger Doll Research para auxiliar en el análisis de los datos recolectados durante la exploración petrolera.

3.2.2. Herramientas para la construcción de Sistemas Expertos

- **CLIPS**: CLIPS [9] es un lenguaje de programación que provee un ambiente de desarrollo para la producción y ejecución de sistemas expertos. Fue creado a partir de 1984, en el Lyndon B. Johnson Space Centerde la NASA. CLIPS es un acrónimo de C Language Integrated Production System (Sistema de Producción Integrado en Lenguaje C). CLIPS probablemente es el motor de reglas más ampliamente usado para la creación de sistemas expertos debido a que es rápido, eficiente y gratuito. Aunque ahora es de dominio público, aún es actualizado y mantenido por su autor original, Gary Riley.

- **Prolog**: Prolog [10] es un lenguaje de programación lógico e interpretado, bastante conocido en el medio de investigación en Inteligencia Artificial.

- **Jess:** Jess [11] es un motor de reglas que incluye dentro de él el lenguaje de programación CLIPS, desarrollado por Ernest Friedman-Hill de los laboratorios Sandia National Labs en 1995 para la creación de sistemas expertos. A continuación será explicado más detenidamente.

3.2.3. Jess

Jess [11] es un motor de reglas y un entorno de programación escrito por completo mediante el lenguaje Java de Sun por Ernest Friedman-Hill de los Laboratorios Nacionales Sandia en Livermore, CA. Este motor de reglas y lenguaje de programación permite la creación de sistemas expertos.

Para usar Jess, se puede especificar la lógica de reglas de dos maneras: mediante el lenguaje de reglas de Jess o mediante XML. Jess también puede proveer algunos de sus propios datos para que las reglas operen. Cuando se ejecute el motor de reglas, éstas se llevarán a cabo. Las reglas crean nuevos datos, o pueden hacer cualquier otra cosa que se pueda hacer mediante la programación en Java.

Aunque Jess se puede ejecutar como un programa autónomo, normalmente se podrán incrustar las librerías de Jess en el código Java y manipularlo usando su propio API de Java o las facilidades básicas ofrecidas por API javax.rules.

Además, Jess puede ser usado para crear servlets de Java, EJBs (Enterprise Java Beans), applets, y un conjunto de aplicaciones que usan el conocimiento en forma de reglas para deducir conclusiones.

Jess usa una versión mejorada del algoritmo Rete [12] para procesar las reglas. El algoritmo Rete es un mecanismo eficiente para resolver problemas de coincidencia de patrones en los sistemas de reglas.

Un sistema experto define una serie de hechos. Cada regla se deberá activar únicamente cuando se den esos hechos. El sistema experto comprobará para cada regla si se cumplen sus hechos comparándolos con la base de conocimientos, activando esa regla en los casos en que se cumplan. Una vez comprobado esto, continúa evaluando la siguiente regla. Este algoritmo, incluso para un número bajo de reglas y hechos, puede tener un tiempo de ejecución muy alto.

El lenguaje de reglas Jess se puede desarrollar en cualquier editor de texto, pero además incluye un entorno de desarrollo basado en la plataforma Eclipse.

Además, Jess posee un interfaz interactivo por línea de comandos que permite ejecutar líneas de código y observar el resultado. De esta manera, se pueden evaluar expresiones matemáticas, ejecutar archivos de código Jess. Este interfaz posee una versión gráfica del interfaz por línea de comandos.

Además, tiene algunas características únicas incluyendo encadenamiento hacia atrás (backward chaining) y memoria de trabajo. Asimismo, puede manipular directamente y razonar acerca de los objetos Java.

3.2.3.1. *Memoria de trabajo*

Cada regla Jess lleva asociada una colección de conocimientos llamados "hechos". Esta colección es conocida como memoria de trabajo. Las reglas Jess únicamente pueden reaccionar a los cambios realizados en la memoria de trabajo.

Todo hecho posee una **plantilla**. Cada hecho se implementa introduciendo los datos en la plantilla. De esta manera, todos los hechos que utilicen la misma plantilla, tendrán los datos almacenados mediante la misma estructura. Cada plantilla es como una tabla de una base de datos en la que las distintas ranuras son las columnas y los distintos registros o filas serían cada uno de los hechos de la memoria de trabajo.

```
(deftemplate template-name
    [extends template-name]
    ["Documentation comment"]
    [(declare  (slot-specific TRUE | FALSE)
               (backchain-reactive TRUE | FALSE)
               (from-class class name)
               (include-variables TRUE | FALSE)
               (ordered TRUE | FALSE))]
     (slot | multislot slot-name
         ([(type ANY | INTEGER | FLOAT |
                NUMBER | SYMBOL | STRING |
                LEXEME | OBJECT | LONG)]
         [(default default value)]
         [(default-dynamic expression)]
         [(allowed-values expression+)])*)
```

Figura 30: Constructor para las plantillas.

En la memoria de trabajo se pueden ver los distintos hechos que existen en cada momento, a qué plantilla pertenecen, los datos que contiene, además de añadir otros nuevos.

Existen dos tipos de hechos:

- Hechos **puros**, aquellos que han sido definidos y creados completamente en Jess. Dentro de este grupo, existen dos tipos de hechos diferenciados:

  - Hechos *desordenados*. Antes de crear hechos desordenados, es necesario definir los distintos campos de la plantilla. La utilidad de los hechos desordenados es que no es necesario crear el hecho y sus campos en el mismo orden en que fueron establecidos en la plantilla. A cada uno de los hechos le es asignado un índice. Se puede borrar un hecho concreto utilizando el índice que se le asigna.

```
Jess> (deftemplate automobile
  "A specific car."
  (slot make)
  (slot model)
  (slot year (type INTEGER))
  (slot color (default white)))
```

Figura 31: Ejemplo de plantilla de un hecho desordenado.

```
(automobile (make Ford) (model Explorer) (year 1999))
```

Figura 32: Ejemplo de un hecho desordenado.

  - Hechos *ordenados*. En algunos casos, los nombres de los campos son redundantes, y fuerzan a escribir más código de lo necesario. Los hechos ordenados son como listas donde el primer campo actúa como el tipo de categoría del hecho. Los hechos ordenados pueden ser añadidos a la memoria de trabajo. Si se añade un hecho cuyo primer campo o cabecera no ha sido usada con anterioridad, Jess asume que se está utilizando un hecho ordenado y crea la plantilla correspondiente automáticamente. Además se puede declarar específicamente una plantilla de hechos ordenados.

```
(shopping-list eggs milk bread)
(person "Bob Smith" Male 35)
(father-of danielle ejfried)
```

Figura 33: Ejemplo de hecho ordenado.

- Hechos **sombra**, aquellos que están relacionados con objetos Java externos. Son similares a los hechos desordenados pero los hechos sombra permiten introducir objetos Java en la memoria de trabajo. Estos hechos actúan como puentes que permiten a Jess razonar sobre cuestiones que ocurren fuera de la memoria de trabajo. Los hechos sombra también necesitan una plantilla, pero ésta se debe indicar como la clase Java a la que se hace referencia en la inicialización de la plantilla.

```
Jess> (deftemplate Account
    (declare (from-class Account)))
```

Figura 34: Ejemplo de hecho sombra

3.2.3.2. *Control de flujo*

La programación basada en Jess soporta control de flujo. La construcción de bucles y el control de flujo están soportados con distintas palabras reservadas. El control de flujo se realiza mediante la llamada a funciones tales como if, while, for, try, etc., que tienen un significado similar a Java u otros lenguajes.

En las figuras siguientes se pueden ver ejemplos de utilización de las funciones while y if respectivamente.

```
Jess> (bind ?i 3)
3
Jess> (while (> ?i 0)
          (printout t ?i crlf)
          (-- ?i))
3
2
1
```

Figura 35: Ejemplo de la utilización de la función while.

En esta figura, se muestra la manera de imprimir una sucesión descendente de números mediante la función *while.*

```
Jess> (bind ?x 1)
1
Jess> (if (> ?x 100) then
          (printout t "X is big" crlf)
       elif (> ?x 50) then
              (printout t "X is medium" crlf)
       else
          (printout t "X is small" crlf))
X is small
```

Figura 36: Ejemplo de la utilización de la función if-else.

En esta otra figura, se muestra la implementación de un sencillo decisor mediante la función if-else.

3.2.3.3. *Variables*

Al igual que el lenguaje de programación Java, Jess soporta la definición de variables. Para ello, el nombre de la variable debe comenzar con el carácter "?", que forma parte del nombre. Las variables no deben ser declaradas antes del primer uso, a excepción de unas variables especiales llamadas variables globales.

Para crear variables que no sean destruidas cada vez que se ejecuta un programa, se utilizan las variables globales creadas mediante el constructor "defglobal". Los nombres de las variables globales, deben comenzar y finalizar mediante el símbolo " * ".

Al definir una variable global es necesario inicializarla a un valor determinado. Cuando se resetea el programa, el valor de la variable vuelve a ser su valor inicial, al menos que se modifique esta propiedad.

A continuación se muestran varios ejemplos de creación de variables tanto globales como no globales.

```
Jess> (bind ?x "The value")
"The value"
```

Figura 37: Creación de una variable de texto.

```
Jess> (bind ?a 123)
123
Jess> ?a
123
```

Figura 38: Creación de una variable numérica.

Como se puede ver en las dos figuras anteriores, no es necesario especificar el tipo de variable que se quiere crear, basta con indicarle el valor al que se quiere inicializar la variable.

```
Jess> (defglobal ?*x* = 3)
Jess> ?*x*
3
Jess> (bind ?*x* 4)
4
Jess> ?*x*
4
Jess> (reset)
Jess> ?*x*
3
```

Figura 39: Creación de una variable global.

Esta última figura muestra la creación de una variable global y cómo al hacer reset del programa, el valor de la variable vuelve a su valor inicial.

3.3. El sonido en Java

Para el tratamiento de material multimedia existen diferentes APIs de Java mejor adaptadas a cada tipo de problemas.

Para el caso del sonido, Java tiene un paquete de librerías específicas, la Java Sound API [13], proporcionado ya con el entorno de desarrollo de Sun. Este paquete cubre diferentes necesidades.

- Permite la captura, el tratamiento, el almacenamiento y la reproducción de audio, tanto en formato MIDI como sonido

muestreado, aunque no se pueden tratar todos los formatos existentes de almacenamiento y transmisión de sonido. Proporciona, además, algún control y efecto sobre el sonido.

- Proporciona las interfaces de síntesis, secuenciamiento y transporte MIDI.
- Proporciona, además, una interfaz para los desarrolladores de servicios basados en las interfaces anteriores.

En el caso particular de la voz, **Java Speech API (JSAPI)** define la interfaz existente para trabajar con sintetizadores y reconocedores de voz. Con JSAPI se puede incorporar la tecnología de la voz en las aplicaciones basadas en tecnología Java.

Además, existe un paquete adicional, que permite capturar, reproducir, transmitir, y codificar múltiples formatos multimedia. Este paquete es **Java Media Framework (JMF)** [14], el cual es una extensión de Java que permite la programación de tareas multimedia en este lenguaje de programación.

Las principales características de JMF son:

- Estabilidad, debido a que funciona sobre la máquina virtual java (JVM).
- Sencillez, ya que permite, usando unos pocos comandos, realizar complejas tareas multimedia.
- Potencia, permitiendo la manipulación de elementos multimedia de vídeo y audio locales (procedentes de la misma máquina en la que se ejecuta el programa), así como la retransmisión en tiempo real de vídeo y audio a través de la red mediante el protocolo RTP.

JMF no se incluye en el paquete básico sino que debe conseguirse como un paquete externo.

3.3.1.jMusic

Además de las tecnologías explicadas hasta ahora, existe otra librería para el tratamiento del sonido mediante Java. **jMusic** [15] es un proyecto

diseñado para proporcionar a los compositores y desarrolladores de software una biblioteca de composiciones musicales y herramientas de procesamiento de audio.

jMusic proporciona una potente estructura para la generación de música, construcción de sonidos de instrumentos, interpretaciones interactivas y análisis musical. jMusic soporta compositores con su propia estructura musical de datos basados en eventos de nota/sonido, y provee métodos para la organización, manipulación y análisis de los datos musicales. Las partituras las devuelve como un archivo MIDI o como un archivo de audio, para su almacenamiento y posterior procesamiento o reproducción en tiempo real. jMusic puede leer y escribir archivos MIDI, archivos de audio, archivos XML y sus propios archivos ".jm".

jMusic permite soporte en tiempo real para JavaSound, QuickTime y MIDIShare. Además, está diseñado para ser extensible para crear composiciones musicales, herramientas e instrumentos propios.

jMusic proporciona un diverso rango de herramientas de visualización y audio como ayuda para ver y oír el estado actual de la composición. Debido a que jMusic es un paquete de Java, ésta se puede servir de los componentes jMusic para escribir aplicaciones.

jMusic es un entorno de composición asistido por ordenador. Está diseñado para facilitar el proceso de composición pero particularmente estructurado para la exploración musical; también puede ser usado para el análisis musical y la educación de la música digital.

A su vez es una API de música. Es una herramienta para sintetizar instrumentos así como para hacer música.

jMusic es libre, es un paquete abierto distribuido bajo la licencia GNU General Public Licence. Ha sido desarrollado por apasionados de la música digital para sus propias creaciones y para otros creadores de música.

jMusic se construye sobre convenciones de la música tradicional occidental. Además se está facilitando la importación y exportación de

archivos MIDI y de audio. Esto significa que los conocimientos musicales anteriores y las herramientas actuales se reutilizan al utilizar jMusic. Asimismo, intenta proporcionar un mayor rango de utilidades de visualización y audición para ayudar a los compositores a ver y oír sus creaciones.

3.3.1.1. *Estructura de datos*

La información musical en jMusic se almacena en una jerarquía basada en las partituras tradicionales de papel.

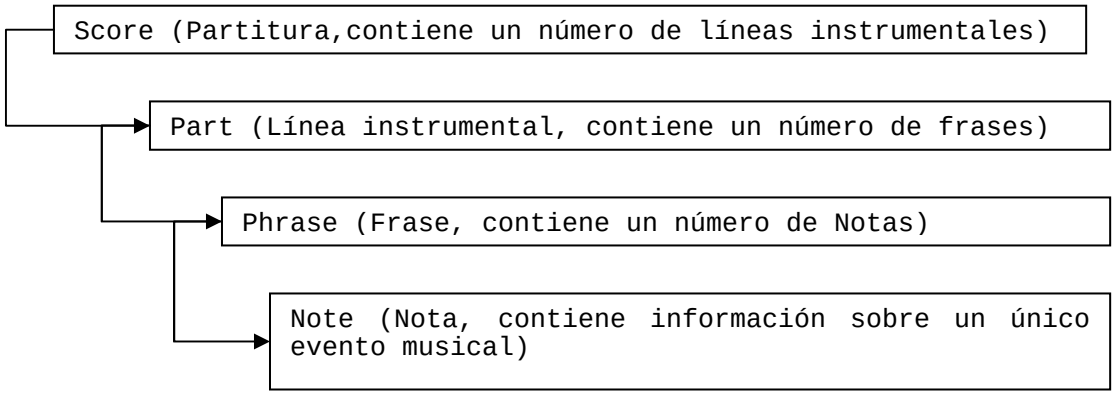


Figura 40: Estructura de datos de jMusic

La nota es la unidad más pequeña. Cada nota se irá almacenando junto con el resto de notas en la siguiente unidad por encima, la frase. Varias frases formarían una parte y varias partes una partitura.

a. *Note*

Contiene información relevante sobre un único evento musical, que a su vez incluye:

- **Frecuencia / Pitch** → Tono de la nota. Altura de la nota. Ejemplo: 440.0 Hz, La, 69 en notación MIDI.
- **Dinámica / Dynamic** →Intensidad. Piano, forte…

- **Figuras Rítmicas / Rhythm Value** → Longitud de la nota, figura. Negra, corchea, redonda…
- **Panorama/ Pan** → Posición de las notas en el espacio estéreo. Izquierda, derecha, centro…
- **Duración / Duration** → Duración de la nota en milisegundos.
- **Offset** → desviación de tiempo desde el comienzo normal de la nota, anacrusa.

Las notas (note) están contenidas en frases (phrase). Al igual que en la notación tradicional, las notas son reproducidas una tras otra según hayan sido incluidas en la frase.

b. *Phrase*

Las frases, musicalmente hablando, se pueden explicar cómo las distintas voces de una partitura. Los objetos Phrase realmente solo contienen un único atributo importante, una lista de notas que pueden ser añadidas, eliminadas o movidas de posición dentro de la frase.

c. *CPhrase*

La clase CPhrase permite al compositor construir estructuras musicales homófonas fácilmente. Se usa de manera similar a Phrase y normalmente se utiliza únicamente para combinaciones complicadas como acordes.

d. *Part*

Contiene las notas suficientes para tocar un instrumento. Cada parte contiene un vector el cual contiene varias Phrase. Estas frases corresponden todas a sonidos similares. Un ejemplo podría ser las distintas voces de los violines en una obra. Una parte tiene un título, un "canal" y un instrumento.

e. *Score*

La clase Score representa el nivel más alto de la estructura de datos., la partitura. Este nivel contiene un vector de partes y un título.

Un objeto Score, contiene una o varias Part, que a su vez puede contener una o varias Phrase, que pueden contener una o varias Note. Esta última puede contener diferentes atributos.

3.3.1.2. *Constantes*

Las constantes son palabras utilizadas por jMusic para referirse a distintos atributos musicales. Existen constantes para la frecuencia, figuras rítmicas, dinámica, timbre, balance, duración articulada, escala, etc.

a. *Frecuencia-Tono / Pitch*

Las notas se escriben en notación tono/octava, de tal manera que el Do central es denominado C4, el Re de esa misma escala, D4 y el Si de la escala anterior B3.

Los accidentes musicales son denominados con S para indicar un sostenido y F para indicar un bemol. Por ejemplo, Do sostenido (Do#) sería CS4.

Las frecuencias se convierten a números enteros para hacer referencia a su valor absoluto al igual que en el estándar MIDI. El rango oscila entre G9-CN1 (Do -1).

A jMusic se le puede especificar directamente la frecuencia de la nota sin necesidad de la constante asociada mediante la sintaxis FRQ[n], donde n es el número de nota asignado según el estándar MIDI. También se puede combinar ambas notaciones, MIDI y constantes de frecuencia, FRQ [C4].

El silencio musical es considerado un caso especial de tono en jMusic. La constante para la ausencia de sonido es REST e indica que ninguna nota debe sonar.

b. *Figuras rítmicas / Rhythmic Value*

Una figura rítmica es un signo que, colocado en un pentagrama con clave establecida, determina la altura y la duración de un sonido determinado.

jMusic está basado en pulsos donde un pulso corresponde al valor 1.0 y el resto de ritmos son relativos a él. Un pulso corresponde a la figura de una negra.

Las constantes relativas a las figuras rítmicas pueden tener varios nombres para denominar una misma constante. Por ejemplo, una negra, es decir, un pulso, se puede denominar mediante las constantes CROTCHET, C, QUARTER_NOTE, y QN. De manera similar, la mitad de un pulso, una corchea, puede ser denominada de diversas maneras, QUAVER, Q, EIGHTH_NOTE, o EN.

Además, también existen constantes para denominar otros ritmos, como DOTTED para añadir medio pulso a la figura, también denominado puntillo en términos musicales. Puede ser aplicado a las distintas figuras. Algunos ejemplos: DOTTED_CROTCHET, DC, CD, DQN, DOTTED_QUARTER_NOTE.

Incluso existen constantes para describir figuras rítmicas que a su vez son conjunto de otras, como los tresillos. Pueden ser aplicadas a distintas figuras, aunque no es normal verlo en todas. Ejemplos: QUAVER_TRIPLET, QT, EIGHTH_NOTE_TRIPLET, ENT.

A continuación se puede ver cómo las figuras rítmicas pueden usarse a la hora de crear un objeto Note y algunos ejemplos de creación de notas:

```
Note n = new Note(frecuencia,figura rítmica );
```

Figura 41: Construcción de un objeto note con sus atributos de frecuencia y figura rítmica.

```
Note n1 = new Note(261.6,CROCHET );
```

Figura 42: Construcción de una nota Do central, de duración un pulso o una negra.

```
Note n2 = new Note(440, Q );
```

Figura 43: Construcción de una nota La central, de duración medio pulso o una corchea.

c. *Dinámica / Dynamic*

La dinámica se refiere a las gradaciones de la intensidad de la música. Existen al menos ocho indicaciones de dinámica, empezando desde un sonido muy quedo, hasta un sonido muy fuerte

jMusic permite establecer la dinámica mediante un valor entero entre 0 y 127. Además, dispone de una serie de constantes que especifican algunos niveles de dinámica usando abreviaturas de términos musicales comunes en italiano.

Tabla 8: Constantes de dinámica.

Abreviatura	**Valor**	**Nombre italiano**
ppp	10	Pianissimo piano o pianississimo
pp	25	Pianissimo
P	50	Piano
mp	60	Mezzo piano
mf	70	Mezzo forte
F	85	Forte
ff	100	Fortisimo
fff	120	Fortissimo forte o fortississimo
Silencio	0	

A continuación se puede ver cómo la dinámica pueden usarse a la hora de crear un objeto Note y algunos ejemplos de creación de notas con dinámica:

```
Note n = new Note(frecuencia, figura rítmica, dinámica);
```

Figura 44: Construcción de un objeto note con sus atributos de frecuencia, figura rítmica y dinámica.

```
Note n = new Note(261.1,CROCHET, F);
```

Figura 45: Construcción de una nota Do central, de duración un pulso o una negra.

```
Note n = new Note(440, Q, P);
```

Figura 46: Construcción de una nota La central, de duración medio pulso o una corchea.

d. *Otras constantes*

Además de las constantes explicadas anteriormente existen algunas otras que pueden ser interesantes en la composición musical. Estas constantes son:

Panoramización (Panning): es la propagación de una señal monoaural en estéreo. El panning permite establecer la panorámica de la imagen estéreo, de tal manera que permite que todo el sonido esté en un solo canal, izquierdo o derecho, o en ambos canales a la vez, central.

Duración Articulada (Duration Articulation): por defecto, la duración de las notas es el 85% del valor de la figura rítmica. Sin embargo se puede especificar una duración diferente multiplicando el valor de la figura rítmica por una constante.

Timbre: el estándar MIDI asigna un número en el rango 1-127 para los distintos instrumentos o timbres más relevantes.

Escala (Scale): define las escalas musicales más utilizadas.

3.3.1.3. *Interfaz*

jMusic proporciona una herramienta para la creación de partituras, visualización y modificación de las partituras ya creadas, etc. Esta herramienta es una interfaz gráfica sencilla que muestra un pentagrama. En la siguiente figura se puede ver la interfaz de jMusic.

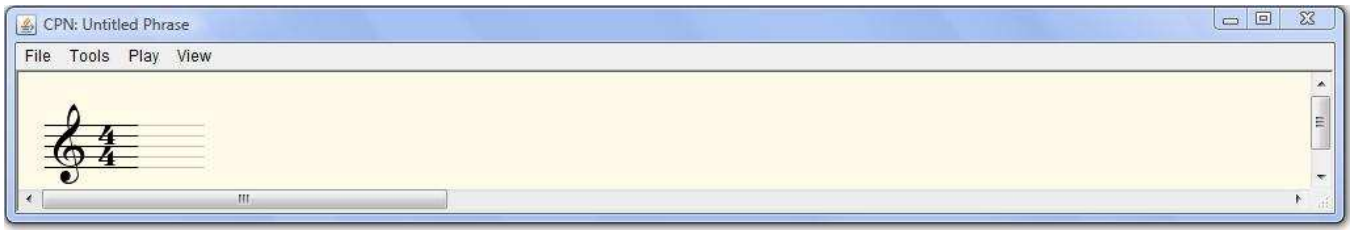


Figura 47: Ejemplo de la interfaz de jMusic

Para crear una nueva partitura, se pueden añadir notas a través del menú *Tools*, explicado más adelante. Una vez añadida una nota, el resto se puede incluir de la misma manera o pulsando encima del pentagrama a la altura que se quiera la nota. Por defecto la interfaz pone como figura una negra. En el caso de que se quiera otra figura se puede modificar colocando el cursor del ratón encima de la nota, pulsando, y moviendo de izquierda a derecha, se modifica la figura.

Lo mismo se puede hacer para modificar la altura de la nota, pero esta vez moviendo de arriba abajo.

Mediante los diferentes menús, se pueden modificar diferentes parámetros y realizar distintas acciones.

El menú *File* permite:

- abrir un nuevo visualizador en blanco, para poder crear una partitura propia
- abrir archivos en los formatos MIDI, XML y .jm, formatos con los que trabaja jMusic
- borrar la última nota o todas las notas
- mostrar u ocultar el tiempo y las alteraciones del compás (el tono)
- además se puede guardar la partitura con otro formato y nombre

El menú *Tools* incluye:

- Set Parameters: permite configurar distintos parámetros de la reproducción como son el *instrumento* con que se quiere que se reproduzca el dictado; el *volumen* al que se quiere que se escuche la reproducción; y el *tempo,* es decir, la velocidad de reproducción de la partitura.
- Add notes by letter: permite añadir notas al final de la partitura referidas mediante la nomenclatura inglesa, la cual nombra a las notas mediante letras.

El menú *Play* permite:

- Play All: reproduce el dictado.
- Repeat All: reproduce el dictado de manera continua varias veces hasta que se presione la opción de Stop Playback.
- Play Last Measure: reproduce el último compás.
- Repeat Last Measure: repite el último compás hasta presionar la opción de Stop.
- Stop Playback: como se ha indicado anteriormente, permite detener la reproducción.

Esto realiza las funciones de un reproductor sin necesidad de utilizar ningún otro software adicional.

El menú desplegable *View*:

- Note data as text: muestra una ventana con los datos de la partitura. La velocidad, el compás, y los datos de las notas (la frecuencia, el ritmo, la duración en segundos y la dinámica).

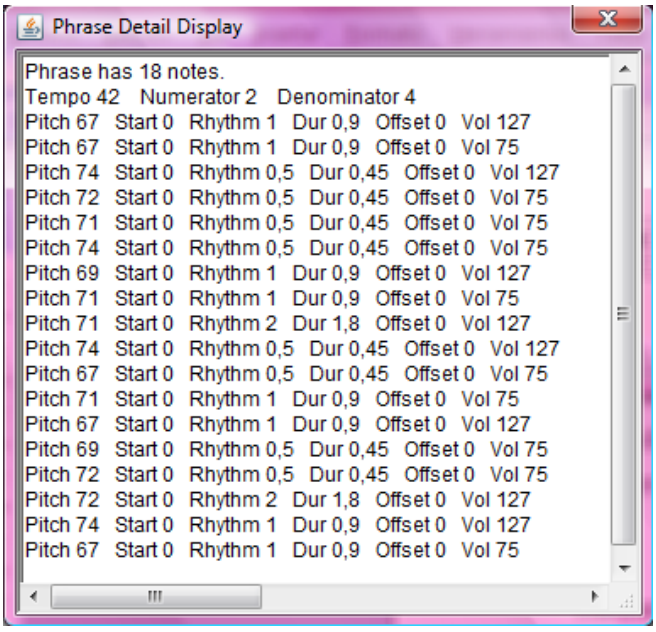


Figura 48: Ejemplo de una partitura en texto plano

- Bar Numbers: permite decidir entre mostrar y no mostrar los números de compás. Por defecto se muestran.

- Stave Title: permite mostrar el título del pentagrama.

3.3.2. Estándares musicales

Existen muchos formatos de archivo de audio digital. Éstos se pueden dividir en dos categorías, formatos PCM y comprimidos. El tamaño de los archivos de audio digital puede depender de la cantidad de canales que tenga el archivo y de la resolución (tasa de muestreo y profundidad).

Formatos PCM. Los formatos PCM [16] poseen toda la información que salió del conversor analógico a digital, sin omitir ningún fragmento y por eso tienen mejor calidad. Dentro de la categoría PCM se encuentran los formatos WAV, AIFF, SU, AU y RAW. La principal diferencia entre estos formatos es el encabezado, cuya duración es de alrededor de 1000 bytes al comienzo del archivo.

Formatos comprimidos. Para usar menos memoria que los archivos PCM existen formatos de sonido comprimidos [16], como por ejemplo el MP3, AAC y Ogg. Ciertos algoritmos de compresión descartan información que no es perceptible por el oído humano para lograr que el mismo fragmento de audio pueda ocupar en la memoria hasta la décima parte de lo que ocuparía de ser PCM. La reducción en tamaño implica una pérdida de información y por esto a los formatos de este tipo se les llama formatos comprimidos con pérdida. Existen también formatos de archivo comprimido sin pérdida, dentro de los que se cuentan el FLAC y el Apple Lossless Encoder, cuyo tamaño suele ser de aproximadamente la mitad de su equivalente PCM.

Formatos descriptivos [16]**:** Este formato de archivos no es precisamente de audio digital, pero sí pertenece a las tecnologías de la informática musical. Este formato no almacena "sonido grabado", sino las indicaciones para que un sintetizador o cualquier otro dispositivo interprete una serie de notas u otras acciones. Por ejemplo, con una partitura, se tendrían con los nombres de los instrumentos que hay que utilizar, las notas, tiempos y algunas indicaciones acerca de la interpretación. El estándar que utiliza este formato es el MIDI.

3.3.2.1. *MIDI*

MIDI son las siglas de *Musical Instrument Digital Interface* (Interfaz Digital de Instrumentos Musicales) [17]. Se trata de un protocolo industrial estándar que permite a los ordenadores, sintetizadores, secuenciadores, controladores y otros dispositivos musicales electrónicos comunicarse y compartir información para la generación de sonidos.

Esta información define diversos tipos de datos como números que pueden corresponder a notas particulares, números de *patches* de sintetizadores o valores de controladores. Gracias a esta simplicidad, los datos pueden ser interpretados de diversas maneras y utilizados con fines diferentes a la música. El protocolo incluye especificaciones complementarias de *hardware* y *software*[18].

Los primeros resultados de esta nueva tecnología se mostraron en el North American Music Manufacturers Show de 1983 en Los Ángeles. La demostración consistió en dos sintetizadores de distintos fabricantes conectados por MIDI con un par de cables; el representante de una de esas dos compañías tocó uno de los sintetizadores los teclados sonaron juntos. Dos instrumentos que soporten el protocolo MIDI pueden comunicarse. La información MIDI tiene un carácter netamente musical: se refiere a comandos play-stop, activación de nota, tempo, volumen, etc, aunque su uso avanzado permite muchas mas posibilidades.

El MIDI es un protocolo de comunicación, un conjunto de comandos que circulan entre dispositivos MIDI dando órdenes a los mismos respecto a lo que deben hacer. La especificación MIDI incluye un aspecto de software que parte de la misma organización de los bytes.

La base de la comunicación MIDI es el byte (una unidad de información digital). Cada comando MIDI tiene una secuencia de bytes específica. El primer byte es el byte de estado (status byte), que le dice al dispositivo MIDI qué función activar. Codificado en ese byte de estado va el canal MIDI. El MIDI opera en 16 canales diferentes, numerados del 0 al 15. Las unidades MIDI aceptarán o ignorarán un byte de estado dependiendo de en qué canal estén configuradas para recibir datos. Sólo este byte de estado

tiene codificado el número de canal, ya que los demás bytes de la cadena se asume que circulan en el canal indicado por el byte de estado.

Tabla 9: Conjunto de bytes de estado más comunes.

Byte Estado	Descripción
1000cccc	Note off
1001cccc	Note on
1010cccc	Postpulsación polifónica
1011cccc	Cambio de control
1100cccc	Cambio de programa
1101cccc	Postpulsación monofónica de canal
1110cccc	Pitch
11110000	Mensaje exclusivo del fabricante
11110001	Mensaje de trama temporal
11110010	Puntero posición de canción
11110011	Selección de canción
11110100	*Indefinido*
11110101	*Indefinido*
11110110	Requerimiento de entonación
11110111	Fin de mensaje exclusivo
11111000	Reloj de temporización
11111001	*Indefinido*
11111010	Inicio
11111011	Continuación
11111100	Parada
11111101	*Indefinido*
11111110	Espera activa
11111111	Reseteo del sistema

Los primeros bytes, cuyos últimos cuatro bits están marcados como "cccc", se refieren a mensajes de canal; el resto de bytes son mensajes de sistema.

Además, dependiendo del byte de estado, le seguirán un número diferente de bytes. Por ejemplo, el estado Note On le dice al dispositivo MIDI que empiece a hacer sonar una nota. Así pues, se requerirán dos bytes adicionales al de estado; uno que indique el tono de la nota (pitch byte) y otro que marque la velocidad de la misma (velocity byte).

Para indicar el tono de la nota correctamente, MIDI asigna un número entero a cada frecuencia estándar de manera que se definen notas particulares. La norma nombra las notas con números que van desde el 0 hasta el 127. El rango de frecuencias que puede reproducir cada instrumento varía según la naturaleza de cada uno.

A continuación se muestra una figura en la que se ve el rango de frecuencias de un piano, el valor MIDI que se le ha asignado y el nombre de la nota correspondiente.

Según se ve, para un piano los valores de las notas van desde la frecuencia 27,500Hz hasta 4186,0Hz. Siendo 440,0Hz la nota de referencia, que corresponde con la nota conocida como La central (número 69 en MIDI) y siendo el Do central, la frecuencia de 261,6 Hz, la nota central (número 60 en MIDI), subiendo o bajando un valor numérico por cada semitono.

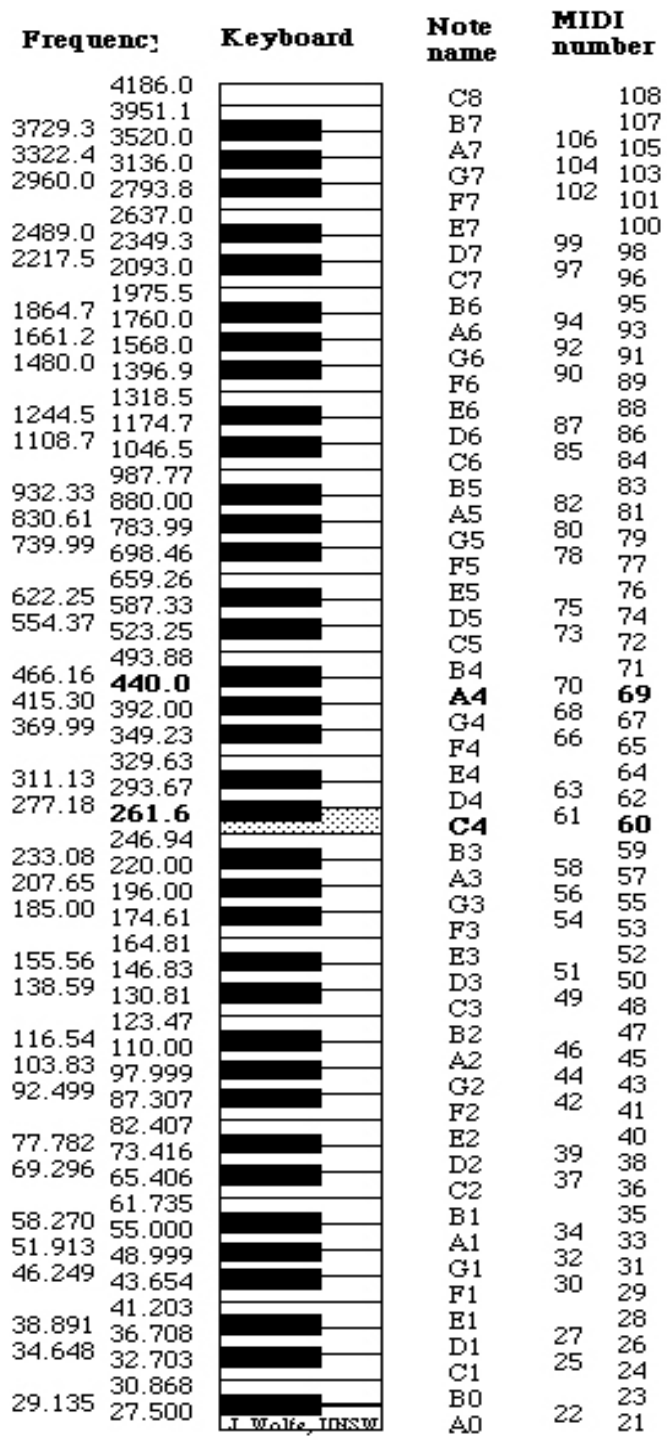


Figura 49: Correspondencia de las notas musicales y las frecuencias en el estándar MIDI.

La imagen que se acaba de presentar es para el caso concreto de un piano, pero MIDI es capaz de reproducir sonidos con diferentes timbres correspondientes a distintos instrumentos. Al igual que para las notas, MIDI especifica hasta 128 instrumentos diferentes. A continuación se pueden ver algunos ejemplos.

Tabla 10: Números asignados a distintos instrumentos por el estándar MIDI.

Numero MIDI	Instrumento o sonido	Numero MIDI	Instrumento o sonido
00	Piano de cola acústico	57	Trombón
10	Caja de música	58	Tuba
12	Xilófono	64	Saxo soprano
21	Acordeón	68	Oboe
22	Armónica	70	Fagot
24	Guitarra española	71	Clarinete
25	Guitarra acústica	73	Flauta
27	Guitarra eléctrica	76	Cuello de botella
29	Guitarra saturada	79	Ocarina
30	Guitarra distorsionada	96	Efecto 1 (lluvia)
31	Armónicos de guitarra	101	Efecto 6 (duendes)
32	Bajo acústico	109	Gaita
40	Violín	122	Playa
41	Viola	123	Piada de pájaro
42	Violoncello	124	Timbre de teléfono
43	Contrabajo	125	Helicóptero
46	Arpa	126	Aplauso
47	Timbales	127	Disparo de fusil
56	Trompeta		

El protocolo MIDI proporciona un formato eficiente y revolucionario para el almacenamiento, edición y reproducción de datos sobre la interpretación musical, mientras que la especificación Standard MIDI File permite que distintas aplicaciones sean capaces de compartir datos MIDI. Además, las escasas exigencias en cuanto a espacio de almacenamiento y la gran facilidad de edición que proporcionan los datos MIDI convierten a este protocolo en un vehículo para la creación musical, edición de partituras, diseño de sonido, sonido en aplicaciones multimedia o juegos de ordenador. El sistema General MIDI suministra unas especificaciones y un conjunto de sonidos comunes que proporcionan a compositores y desarrolladores de aplicaciones multimedia una plataforma común y estandarizada.

4. DISEÑO DEL SISTEMA

En este capítulo se explica en primer lugar la funcionalidad global del sistema y de los módulos que lo componen. Posteriormente se describirá detenidamente cada módulo por separado.

4.1. Funcionalidad del sistema

El sistema se ha diseñado con una arquitectura modular que se presenta en la Figura 50. Este sistema está formado por cuatro módulos diferentes: Interfaz de Usuario de Entrada, Interfaz de Usuario de Salida, Sistema Experto y Constructor.

Los dos primeros módulos, correspondientes con las interfaces de usuario de entrada y salida, serán los encargados de interactuar con el usuario.

Al comenzar la ejecución del sistema, la interfaz de entrada mostrará distintas opciones entre las que el usuario tendrá que elegir. El sistema almacena estas opciones. Estas opciones son el tipo de dictado y el nivel de dificultad del mismo, según se describirá más adelante.

El segundo módulo es el denominado Sistema Experto. Este módulo será el que realmente genere el dictado. A partir de los parámetros elegidos por el usuario, en base a unas reglas musicales establecidas de antemano y a partir del comportamiento real de un experto humano, este segundo módulo decidirá los parámetros/componentes del dictado, esto es, definirá los posibles tiempos, el conjunto de notas adecuado, etc. A partir de este punto, el sistema tendrá que dar forma a esos argumentos y crear la partitura.

El tercer módulo es el denominado Constructor. Este módulo incluye distintas tecnologías para crear el resultado final y mostrárselo al usuario. Es decir, dará forma a los parámetros devueltos por el módulo anterior.

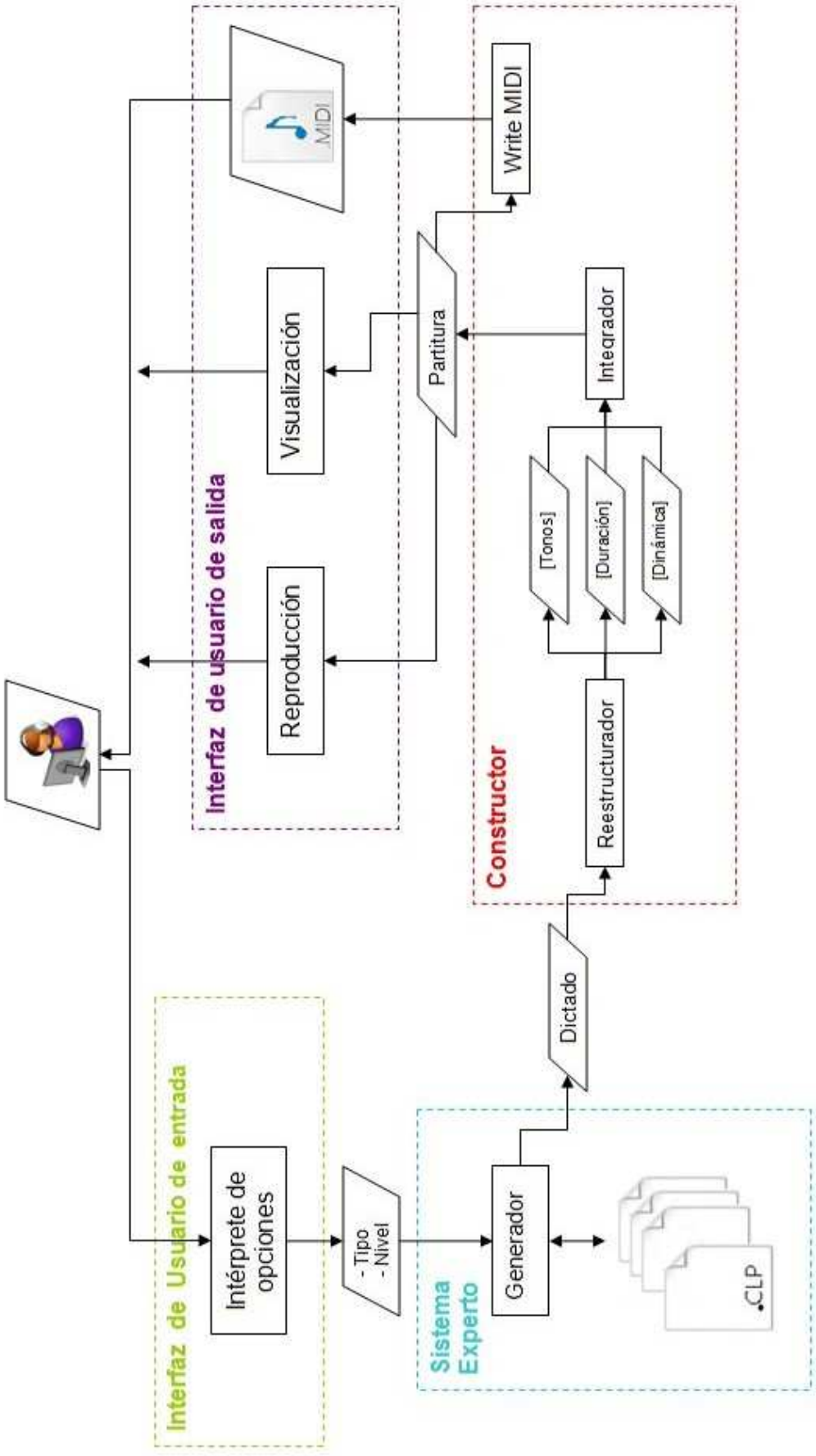


Figura 50: Diagrama del Sistema

Este módulo es el encargado de estructurar los datos referentes al dictado de manera que el sistema pueda presentarlos al usuario. También tiene como función crear la partitura, es decir, agrupar los datos en unidades pequeñas primero (figuras), y en unidades mayores a continuación (compases), de manera que al final se obtenga la partitura. Con esta partitura se creará un archivo MIDI.

Al final del proceso de creación del dictado, el módulo de la interfaz de salida será en encargado de mostrar de diferentes maneras los resultados obtenidos, visual o auditivamente. Por ejemplo, auditivamente mediante un reproductor propio del sistema o visualmente representado de forma gráfica la partitura en un pentagrama.

Los tipos de dictado que ofrece este sistema son rítmicos o melódicos.

- Dictado Rítmico: dictado cuyo objetivo es desarrollar la capacidad de diferenciar entre las diferentes figuras musicales. Es por eso que lo único relevante son las figuras musicales, el ritmo. El tono de las notas no varía y tampoco la tonalidad del dictado. Son algo menos completos que los melódicos, pero permite aprender lo referente al ritmo de manera más sencilla.

- Dictado Melódico: Dictado cuya finalidad es entrenar el oído tanto en relación al ritmo como en las distintas notas. Así, incluye diferentes notas, ritmos, tonalidades, compases, etc. Es más completo que el rítmico y a su vez más complejo.

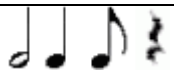
Los niveles del dictado dependen del si el dictado es rítmico o melódico.

Niveles Rítmicos

En los niveles rítmicos se tendrán en cuenta los parámetros referentes al tiempo del compás y las figuras, explicados en los apartados 2.1 y 2.2. En la Tabla 11 se detallan las diferentes configuraciones para

cada uno de los niveles, seleccionadas según la dificultad real percibida por un estudiante de música.

Tabla 11: Descripción de Niveles Rítmicos.

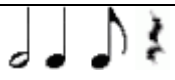
	Tiempo	**Figuras posibles**
Nivel 0	2/4	
Nivel 1	2/4	
Nivel 2	2/4	
Nivel 3	3/4	
Nivel 4	3/4	

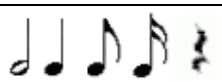
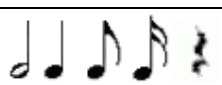
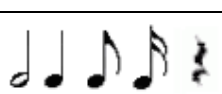
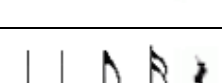
Al no ser relevante el tono de las notas, los niveles no se diferenciarán por la tonalidad.

Niveles Melódicos

En los dictados melódicos, se tendrán en cuenta los parámetros Tonalidad, Alteraciones, Tiempo, Notas y Figuras rítmicas, comentados en el Capítulo 2. En la Tabla 12 se detallan las diferentes configuraciones para cada uno de los niveles.

Tabla 12: Niveles melódicos.

Nivel	**Tonalidad**	**Alteraciones**	**Tiempo**	**Notas**	**Figuras**
Nivel 0	Do M	0	2/4	I / V	
Nivel 1	Do M	0	2/4	I/III/V	
Nivel 2	Do M	0	2/4	I – V	
Nivel 3	Do M	0	2/4	I – V	

Nivel 4	Do M	0	2/4	I – VII	
Nivel 5	Sol M	Fa#	2/4	I/III/V	
Nivel 6	Sol M	Fa#	2/4	I – V	
Nivel 7	Sol M	Fa#	2/4	I – VII	
Nivel 8	Re M	Fa#, Do#	2/4	I – VII	
Nivel 9	La M	Fa#, Do#, Sol#	2/4	I – VII	
Nivel 10	Fa M	Si b	2/4	I – VII	
Nivel 11	Do M	0	3/4	I – VII	

A continuación se explicará detenidamente cada uno de los módulos y los distintos componentes de los mismos.

4.2. Interfaz de Usuario de Entrada

Es el encargado de interactuar a la entrada con el usuario y guardar los parámetros elegidos por éste.

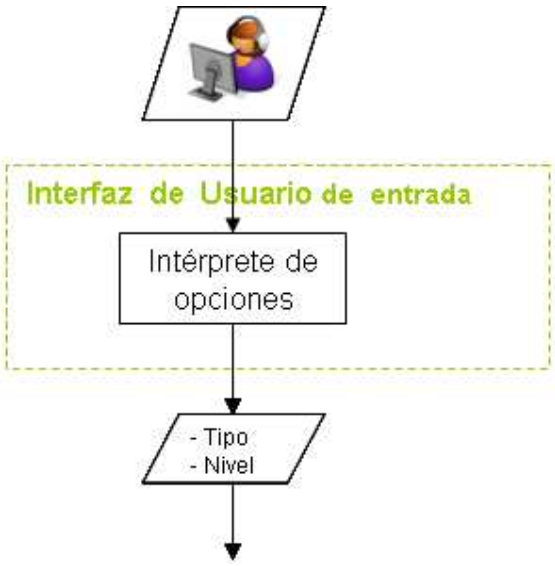


Figura 51. Diagrama del interfaz de usuario de entrada.

4.2.1. Intérprete de opciones

El *intérprete de opciones* permite al usuario elegir entre varias posibilidades para generar el dictado. Las opciones que el usuario podrá elegir son:

- **Tipo de dictado**: R (rítmico) o M (melódico).
- **Nivel del dictado**. Dependiendo de si el dictado es rítmico o melódico, el número de niveles de dificultad es distinto. Para los dictados rítmicos el número de niveles es de 0 a 4 y para los dictados melódicos es 0 a 11.

La manera de interactuar con el usuario es a través de la consola y el teclado. Las opciones se muestran a través de la consola y el usuario introduce las respuestas a través del teclado.

```
Elija el tipo de dictado, Melódico(M) o Rítmico(R):

Elija el nivel del dictado de 0 a _:
```

Figura 52: Opciones mostradas al usuario.

Estos datos son guardados en el sistema para que el módulo del Sistema Experto pueda crear el dictado en función de estos parámetros.

4.3. Sistema Experto

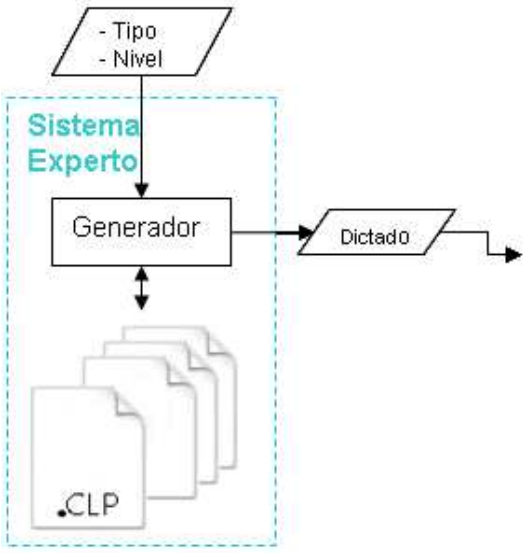


Figura 53: Diagrama del módulo del Sistema Experto.

El sistema experto es el módulo que, a partir de los datos guardados por el intérprete de opciones y unas reglas básicas de composición de dictados, generará la información sobre el dictado.

Esta información se compone de cinco atributos del dictado que se pueden dividir en dos grupos. Una de las divisiones son aquellos atributos específicos de cada nota. El sistema debe elegir entre varias opciones cual es el valor más adecuado para cada uno de ellos:

- Tono de la nota.

- Duración de la nota.

- Dinámica de la nota.

Los otros dos atributos, no son específicos de cada nota, pero son necesarios para la construcción del dictado. Para estos atributos se va calculando cuál es su valor, no se elige entre varios:

- Número del compás.

- Número de nota.

Para los dictados rítmicos, el tono de la nota no sería necesario. Para los dictados melódicos serían necesarios todos los atributos.

El sistema experto genera una secuencia de notas las cuales poseen una serie de atributos, que cumplen con las especificaciones indicadas, en función de la configuración especificada por el usuario y el conjunto de reglas definido en el motor de reglas. El sistema experto va almacenando la información en la memoria de trabajo del motor de reglas en forma de hechos de manera que cada hecho hace referencia a una nota. La salida de este módulo sería el conjunto de hechos almacenados en la memoria de trabajo.

4.4. Constructor

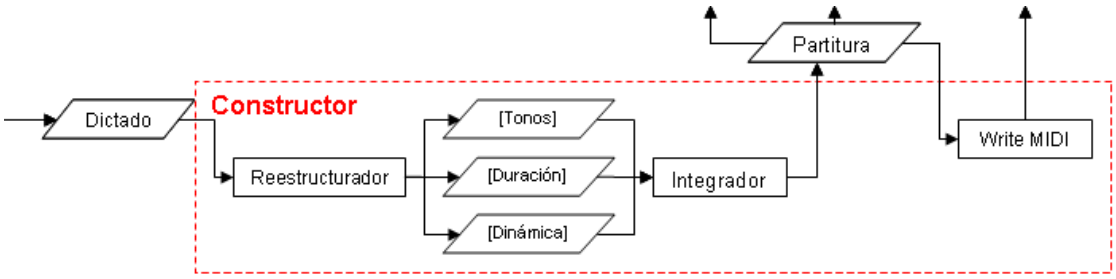


Figura 54: Diagrama del módulo del Constructor.

El módulo *Constructor* tiene como entrada la información almacenada por el módulo anterior en la memoria de trabajo del motor de reglas. Es el encargado de recuperar los datos almacenados, interpretarlos y modificarlos para construir la partitura. Esto será realizado a su vez por dos submódulos.

4.4.1. Reestructurador

Se encarga de recuperar la información almacenada en la memoria de trabajo del sistema experto, reordenarla y presentarla de nuevo en forma de vectores.

De todos los atributos guardados en la memoria de trabajo para cada nota descritos en el apartado anterior, el módulo *Reestructurador* únicamente tomará los necesarios para la construcción de la partitura. Concretamente, serán los campos del tono de la nota, su longitud y su dinámica con los que trabajará. Cada nota queda completamente definida con estas tres características.

El resultado de este módulo es la construcción de tres vectores, uno por atributo. Cada vez que se recupere un hecho, la información relativa a cada atributo es almacenada en un vector diferente. Por lo tanto, existirá un vector para las frecuencias, otro vector para las longitudes y por último, otro vector para la dinámica de la secuencia de notas generada por el compositor de dictados.

4.4.2. Integrador

Una vez se tiene la información extraída de la memoria de trabajo y guardada en los tres vectores, se puede encapsular en pequeñas unidades musicales. Esta unidad es la denominada por jMusic *Note*. *Note* hace referencia no únicamente al tono de la nota, sino a los tres atributos, tono, duración y dinámica.

Este módulo se encarga de ir extrayendo la información de los vectores e ir encapsulándola en cada una de las notas. De esta manera se construye la partitura a partir de los tres vectores de información proporcionados por el módulo *Reestructurador*. Esta partitura estará formada por el conjunto de todas las frecuencias de las notas, sus duraciones, y sus dinámicas. El resultado de este módulo es partitura del dictado en forma de texto plano. Como se puede ver en la Figura 55, el silencio está representado por un número negativo de valor muy elevado. De esta manera se puede diferenciar fácilmente de cualquier otro tono. En este caso, la dinámica no tiene relevancia.

```
-------- jMusic PHRASE: 'Dictado' contains 26 notes. --------
jMusic NOTE: [Pitch = 60][RhythmValue = 0.25][Dynamic = 127]
jMusic NOTE: [Pitch = 72][RhythmValue = 0.25][Dynamic = 65]
jMusic NOTE: [Pitch = 69][RhythmValue = 0.25][Dynamic = 65]
jMusic NOTE: [Pitch = 69][RhythmValue = 0.25][Dynamic = 65]
jMusic NOTE: [Pitch = 64][RhythmValue = 1.0][Dynamic = 65]
jMusic NOTE: [Pitch = 64][RhythmValue = 0.5][Dynamic = 127]
jMusic NOTE: [Pitch = 69][RhythmValue = 0.5][Dynamic = 65]
jMusic NOTE: [Pitch = 64][RhythmValue = 0.5][Dynamic = 65]
jMusic NOTE: [Pitch = 65][RhythmValue = 0.5][Dynamic = 65]
jMusic NOTE: [Pitch = 65][RhythmValue = 0.5][Dynamic = 127]
jMusic NOTE: [Pitch = 62][RhythmValue = 0.5][Dynamic = 65]
jMusic NOTE: [Pitch = 72][RhythmValue = 0.5][Dynamic = 65]
jMusic NOTE: [Pitch = 62][RhythmValue = 0.5][Dynamic = 65]
jMusic NOTE: [Pitch = 67][RhythmValue = 2.0][Dynamic = 127]
jMusic NOTE: [Pitch = -2147483648][RhythmValue = 1.0][Dynamic = 127]
jMusic NOTE: [Pitch = 62][RhythmValue = 0.25][Dynamic = 65]
jMusic NOTE: [Pitch = 67][RhythmValue = 0.25][Dynamic = 65]
jMusic NOTE: [Pitch = 69][RhythmValue = 0.25][Dynamic = 65]
jMusic NOTE: [Pitch = 71][RhythmValue = 0.25][Dynamic = 65]
jMusic NOTE: [Pitch = 64][RhythmValue = 2.0][Dynamic = 127]
jMusic NOTE: [Pitch = 67][RhythmValue = 0.5][Dynamic = 127]
jMusic NOTE: [Pitch = 71][RhythmValue = 0.5][Dynamic = 65]
jMusic NOTE: [Pitch = 60][RhythmValue = 0.5][Dynamic = 65]
jMusic NOTE: [Pitch = 69][RhythmValue = 0.5][Dynamic = 65]
jMusic NOTE: [Pitch = 67][RhythmValue = 1.0][Dynamic = 127]
jMusic NOTE: [Pitch = 60][RhythmValue = 1.0][Dynamic = 65]
```

Figura 55: Partitura generada por el módulo Integrador en forma de texto plano.

4.4.3. WriteMidi

Una vez que el sistema ha creado la partitura con la información relativa a las notas, se da lugar a la construcción de un archivo de audio en formato MIDI en base a la partitura e indicando el nombre que se le quiere dar al archivo MIDI.

Una vez construido el archivo MIDI, el usuario podrá reproducirlo con un reproductor de audio que soporte este formato. Además, podrá realizar todas las opciones que dicho reproductor le permita como parar el dictado, rebobinarlo, etc.

4.5. Interfaz de Usuario de Salida

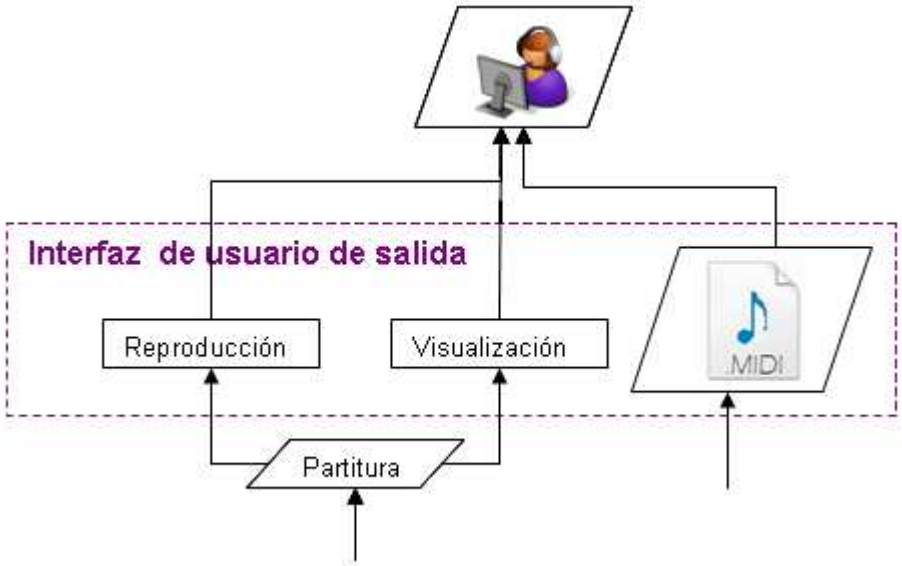


Figura 56: Diagrama del módulo de la Interfaz de Usuario de Salida.

La Figura 55, muestra toda la información necesaria para el dictado, pero no es comprensible por el usuario, por lo que hay que procesarla de manera que éste pueda interpretarla.

Una vez que el sistema ha generado la información y la ha almacenado dentro de un objeto *Score* o partitura, se presentan al usuario distintas opciones.

```
Para escuchar el dictado, pulse la tecla E. Para ver el dictado, pulse V:
```

Figura 57: Opciones mostradas al usuario.

Si la tecla pulsada es la "E", el dictado empezará a sonar sin mostrar ningún tipo de interfaz gráfica mediante el Reproductor. Si se pulsa la tecla "V", se visualizará la representación gráfica de la partitura mediante la Interfaz de Visualización.

4.5.1. Reproductor

Un dictado musical es un ejercicio auditivo en el cual el usuario debe transcribir la música que está escuchando. Es por ello necesario que la partitura generada pueda ser reproducida y escuchada por parte del usuario.

Esto es posible gracias a que jMusic proporciona un reproductor propio, de manera que el dictado es oído pero no es visto. El reproductor es capaz de generar los sonidos a partir de los datos guardados en la partitura creada anteriormente. Para reproducir el dictado será necesario disponer de unos altavoces conectados a la salida de la tarjeta de audio del ordenador.

4.5.2. Interfaz de Visualización

En música, como ya se ha explicado en el apartado 2.1, la representación gráfica de los sonidos se hace por medio de las figuras musicales, que se escriben sobre la pauta o pentagrama.

jMusic proporciona unas funciones para la visualización de las partituras creadas. A partir de la partitura (score) creada como conjunto notas, jMusic imprime los datos transformándolos gráficamente en los símbolos musicales.

El motivo de dar la oportunidad al usuario de visualizar el dictado sobre el pentagrama es que éste pueda comprobar si lo que ha creído escuchar es realmente el dictado que ha sonado.

En la figura siguiente se puede ver un ejemplo de un dictado representado gráficamente sobre un pentagrama.

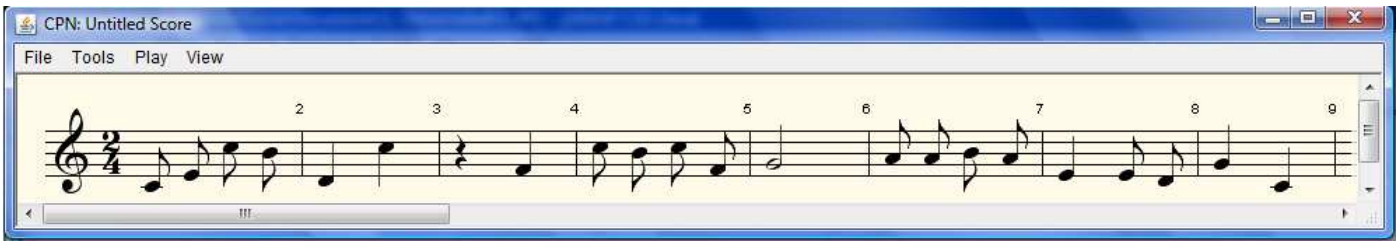


Figura 58: Visualización de un dictado.

Como se ve, en el pentagrama se indica la clave, de Sol; el tiempo del compás, 2/4; y las notas que corresponden al dictado generado que corresponde al nivel 4.

5. IMPLEMENTACIÓN

La implementación de este sistema ha sido desarrollada en Java integrando dos tecnologías, el motor de reglas Jess y la librería de Java jMusic.

5.1. Interfaz De Usuario de Entrada

La interfaz de usuario de entrada es una sencilla interfaz desarrollada mediante Java. Por pantalla se le muestran al usuario las distintas opciones y éste debe introducirlas mediante teclado. El sistema lee de la entrada estándar las opciones elegidas y las almacena. La lectura de teclado se realiza mediante un método creado para tal efecto basada en el método propio de Java readLine() el cual lee una línea de texto. Los parámetros almacenados serán los que utilizarán posteriormente el resto de módulos.

5.2. Sistema Experto

El módulo del Sistema Experto ha sido desarrollado en su mayor parte mediante el motor de reglas Jess. Una pequeña parte está desarrollada mediante jMusic. Como se explicado en el apartado 3.3.3, para usar Jess, se puede especificar la lógica de reglas de dos maneras, mediante el lenguaje de reglas de Jess o mediante XML. Para el desarrollo de este sistema, se ha elegido la primera opción, el lenguaje de reglas Jess.

Dependiendo de los parámetros del dictado que el usuario elija, se cargará en la base de conocimientos y luego se ejectará un conjunto de reglas u otro. La manera elegida para ejecutar un conjunto de reglas u otro es almacenando el conjunto de reglas referente a cada nivel de cada tipo de dictado en un fichero de texto diferente. Además, existen dos archivos principales, uno para los rítmicos y otro para los melódicos, en los cuales se decide qué archivo con un determinado conjunto de reglas se deberá ejecutar. Concretamente, indican que se debe ejecutar uno u otro conjunto de reglas dependiendo del nivel.

Con la elección del tipo de dictado, el sistema, mediante la librería jMusic y el algoritmo Rete, ejecuta el archivo principal correspondiente, "dictadoM.clp" o "dictadoR.clp".

```
Rete rete= new Rete();

String archivoExt = "dictado" + tipo + ".clp";

rete.batch(archivoExt);
rete.run();
```

Figura 59. Código de ejecución del archivo que define el tipo de dictado.

Como se puede observar en la Figura 59, si por ejemplo el usuario elige dictados rítmicos (R), el algoritmo rete, el cual nos permite ejecutar código de Jess desde la clase Java, ejecutará el archivo "dictadoR.clp" mediante la sentencia *batch,* que lleva a cabo la ejecución.

Pero además del tipo de dictado, también interviene el nivel del mismo. El nivel se le indica al motor de reglas Jess desde la clase Java, ya que es aquí donde se almacena la elección del usuario. Este tipo de datos se definen en forma de conjunto de hechos iniciales en Jess, "deffacts".

```
String deffactInicial = "(deffacts inicial\n" +
                        "(posicion primera)"+
                        "(nivel "+ nivel +")\n" + ")";
```

Figura 60. Conjunto de hechos iniciales en función del nivel del dictado.

Como se puede observar en la Figura 55, además del nivel, también se está indicando la posición de la nota. En todos los casos, las reglas diferenciarán si la nota es la primera del dictado, si es la última o si es una nota intermedia. Es decir, las reglas tomarán diferentes decisiones en función de la posición, el tipo de dictado y el nivel. Como antes de empezar el dictado la nota a crear es la primera, la posición que se indica es la primera.

Es necesario que estos hechos iniciales sean evaluados antes de ejecutar el archivo referente al tipo de dictado, explicado anteriormente. En la Figura 61 se puede ver la secuencia ordenada de ejecución del código.

```
//El sistema pregunta distintas opciones al usuario y lee por teclado las respuestas
System.out.println("\n Elija el tipo de dictado, Melodico(M) o Ritmico(R): \n ");

//lee y almacena el tipo de dictado
String tipo = readStr();

//Nombre del archivo Jess a ejecutar
String archivoExt = "dictado" + tipo + ".clp";

//Da a elegir el nivel del dictado
if (tipo.equals("M")){
   System.out.println("\n Elija el nivel del dictado del 0 al 10: \n");
}
else if (tipo.equals("R")){
   System.out.println("\n Elija el nivel del dictado del 0 al 4: \n");
}
else
   return false;

//lee y almacena el nivel del dictado
nivel = readStr();


//Crea los hecho iniciales
deffactInicial = "(deffacts inicial\n" +
                 "(posicion primera)"+
                 "(nivel "+ nivel +")\n" + ")";


//Carga los hechos iniciales
rete.eval(deffactInicial);
rete.reset();
//Ejecutamos el programa JESS
rete.batch(archivoExt);
rete.run();
```

Figura 61. Secuencia ordenada de ejecución.

Una vez que se ha ejecutado el código Jess correspondiente al tipo de dictado, son las reglas Jess las que deciden el siguiente código a ejecutar. Dependiendo del nivel, se indica el nuevo archivo que el motor Jess deberá ejecutar. Los archivos están divididos por niveles. Así, existen diecisiete archivos diferentes, cinco para los cinco niveles de los dictados rítmicos y doce para los melódicos. En la Figura 62 se puede ver un ejemplo de código donde se decide el conjunto de reglas que hay que ejecutar para crear el dictado dependiendo del nivel.

```
(defrule nivel0
    (nivel 0)
=>
    (batch "Melod0.clp")
    (retract-string "(nivel 0)")
)

(defrule nivel1
    (nivel 1)
=>
    (batch "Melod1.clp")
    (retract-string "(nivel 1)")
)

(defrule nivel2
    (nivel 2)
=>
    (batch "Melod2.clp")
        (retract-string "(nivel 2)")
)
```

Figura 62: Ejecución de las reglas de decisión del nivel.

La primera de las reglas de la Figura 62 establece que si `nivel` es `0`, el archivo a ejecutar sería "Melod0.clp". Además, hay que borrar de los hechos el valor de `nivel` para la siguiente ocasión que se cree un dictado.

Las reglas segunda y tercera establecen lo mismo que la anterior, solo que para valores de `nivel` 1 y 2. Esto continuaría con el resto de niveles tanto para dictados rítmicos como melódicos.

5.2.1. Descripción de las reglas

Como ya se ha comentado anteriormente, es necesaria la definición de una serie de reglas para que el motor de reglas sea capaz de generar el dictado. Estas reglas hacen referencia tanto al tono de las notas, su duración y su dinámica. Han sido desarrolladas mediante Jess.

Antes de definir las distintas reglas, es necesario definir una serie de variables que se usarán posteriormente pero cuyo valor debe ser el mismo al comenzar la ejecución. Para ello se utilizan las variables globales, explicadas en el Capítulo 3.3.2.

```
(defglobal
  ?*Tonica* = 60
  ?*Segunda* = (+ ?*Tonica* 2)
  ?*Tercera* = (+ ?*Tonica* 4)
  ?*Cuarta* = (+ ?*Tonica* 5)
  ?*Dominante* = (+ ?*Tonica* 7)
  ?*Sexta* = (+ ?*Tonica* 9)
  ?*Septima* = (+ ?*Tonica* 11)
  ?*Octava* = (+ ?*Tonica* 12)
  ?*Silencio* = -2147483648
  ?*numCompas* = 1
  ?*base* = 4
  ?*restanteCompas* = 2
  ?*contador* = 0
  ?*minima* = 1
  ?*posicionAnterior* = 0)
```

Figura 63: Conjunto de variables globales.

En la definición de las variables globales, se puede ver que se define cuál va a ser la tónica y a partir de ahí, se calculan el resto de grados de la escala. Además, se define el valor del silencio, un número muy grande negativo. El número de compás inicialmente es el uno, restanteCompas es el número de pulsos sin asignar nota que tiene cada compás. Cada pulso tiene el valor que indica base. El resto de variables son variables auxiliares utilizadas a lo largo del proceso.

5.2.1.1. *Reglas sobre el tono de la nota*

Para todas las notas de los dictados melódicos, el sistema debe decidir qué tono va a tener la siguiente nota. Esta elección se realiza de manera aleatoria excepto en dos casos especiales. Para elegir aleatoriamente el tono, se utiliza la función `nota()`. Esta función elige aleatoriamente entre una serie de tonos dados. Dependiendo del nivel del dictado, el número de tonos posible varía. La función genera un número aleatorio y dependiendo del número la función devuelve un tono u otro. El silencio también es entendido como un posible tono.

```
(deffunction nota()

    (bind ?n (mod (random) 5))

    (if (= ?n 0) then
        (return ?*Tonica*)
    else
        (if (= ?n 1) then
            (return ?*Segunda*)
        else
            (if (= ?n 2) then
                (return ?*Tercera*)
            else
                (if (= ?n 3) then
                    (return ?*Cuarta*)
                else
                    (if (= ?n 4) then
                        (return ?*Dominante*)
                    )
                )
            )
        )
    )
)
```

Figura 64: Función nota del nivel melódico 2.

Las dos restricciones que se han tenido en cuenta a la hora de elegir las notas corresponden a la primera y dos últimas notas de cada dictado. La primera restricción es que la primera nota del dictado debe ser la tónica o grado I de la tonalidad elegida. La tonalidad dependerá del nivel. Las diferentes tonalidades posibles de este sistema son Do M, Sol M, Re M, La M y Fa M ya que poseen el número de alteraciones adecuado a los niveles establecidos. Sus tónicas son Do, Sol, Re, La y Fa respectivamente.

```
(bind ?nota ?*Tonica*)
```

Figura 65: Código que almacena la tónica como tono de la nota.

La segunda de las restricciones es que las dos últimas notas del dictado deben corresponder a la cadencia dominante - tónica o grados V - I de la tonalidad. Esto quiere decir que, por ejemplo, el dictado cuya tonalidad es Do M, debe finalizar con las notas Sol y Do correspondiendo el Sol a la penúltima nota y el Do a la última según la cadencia musical

perfecta o auténtica. A continuación se muestran las notas correspondientes a los grados I y V para cada una de las tonalidades.

```
(bind ?nota ?*Dominante*)
(bind ?nota ?*Tonica*)
```

Figura 66: Código que almacena la decadencia dominante-tónica.

Tabla 13: Dominante y Tónica de las tonalidades del dictado.

Tonalidad	I Grado	V Grado
Do M	Do	Sol
Sol M	Sol	Re
Re M	Re	La
La M	La	Mi
Fa M	Fa	Do

Además, existe otra pequeña restricción respecto a la altura de los tonos. Si las notas tienen un tono muy agudo o grave respecto a la clave de Sol se dificulta la lectura de las partituras. Por ello se ha optado por que los tonos no sobrepasen por superior o inferiormente determinados tonos. De esta manera, si alguna nota superase esos límites, la nota pasaría a tener el mismo tono pero una octava inferior, esto es, restando 12 semitonos.

```
(if (> ?nota2 74) then
    (bind ?nota2 (- ?nota2 12))
else
    (if (< ?nota2 60) then
        (bind ?nota2 (+ ?nota2 12))
    )
)
```

Figura 67: Restricción de la altura del tono.

Figura 68: Ejemplo de dictado melódico (I).

En el ejemplo de la Figura 68 se pueden ver la aplicación de las normas explicadas anteriormente respecto al tono. Este dictado está en Sol

mayor, por lo que la primera nota es un Sol y las dos últimas cumplen la cadencia perfecta V-I, es decir, Re – Do. Además, se puede ver que se cumple la restricción de la altura del tono ya que ninguna de las notas está por debajo del Do4 ni por encima del Re5.

5.2.1.2. *Reglas sobre la duración de la nota*

Para todas las notas de los dictados tanto melódicos como rítmicos, el sistema debe decidir qué figura o longitud va a tener la siguiente nota. Esta elección se realiza de manera aleatoria. Para ello, utiliza la función `longitud()`. Esta función elige aleatoriamente entre una serie de longitudes posibles. Dependiendo del nivel del dictado, el número de opciones varía.

```
(deffunction longitud(?restante)
    (bind ?n (mod (random) 6))
    (if (or (= ?n 0) (= ?n 1)) then
        (bind ?l 1)
    else
        (if (or (= ?n 2) (= ?n 3)) then
            (bind ?l 0.5)
        else
            (if (= ?n 4) then
                (bind ?l 0.25)
            else
                (if (= ?n 5) then
                    (bind ?l 2)
                )
            )
        )
    )
    (if (> ?l ?restante) then
        (return ?restante)
    else
        (return ?l)
    )
)
```

Figura 69: Función longitud del nivel rítmico 2.

Al igual que la función `nota()`, esta función genera un número aleatorio y dependiendo de éste, se devuelve una longitud de figura u otra. Como cada compás tiene una capacidad limitada, esta función comprueba

que el valor de la figura generada no es mayor que la capacidad restante. Si supera esta capacidad, la figura pasa a tener el valor de la capacidad que falte por llenar.

Respecto a la duración de la nota, las reglas verifican si la suma de las duraciones de ese compás es igual al total permitido para cada tiempo del compás. Si es así, pasan al siguiente compás. Además, las únicas combinaciones posibles de figuras serán de dos corcheas o de cuatro semicorcheas por cada pulso de negra. Es decir, en los dictados creados por este sistema no existirán síncopas, ligaduras ni combinaciones de figuras diferentes a las mencionadas ya que dan una dificultad mayor que la tratada en este proyecto.

```
(bind ?*restanteCompas* (- ?*restanteCompas* ?longitud))
(if (= ?*restanteCompas* 0) then
    (bind ?*numCompas* (+ ?*numCompas* 1))
    (bind ?*restanteCompas* 2)
)
```

Figura 70: Código que cambia de compás una vez se llega al límite.

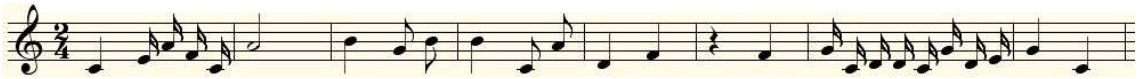


Figura 71: Ejemplo de dictado melódico (II).

En la figura anterior, Figura 71, se puede ver que en todos los compases, la suma de las distintas duraciones es igual al total de lo permitido. Al ser un 2/4, cada compás puede albergar una blanca, dos negras, cuatro corcheas, ocho semicorcheas o combinaciones de éstas.

5.2.1.3. *Reglas sobre la dinámica de la nota*

La decisión sobre la dinámica, es decir, si una nota suena *mezzopiano o forte,* se realiza mediante reglas que comprueban si la nota es la primera del compás o no. En el caso de que sea la primera nota tocada en el compás, deberá sonar con más intensidad, es decir *forte*. El resto de notas del compás, deberán sonar con menor intensidad, es decir, *mezzopiano.*

```
(if (= ?*restanteCompas* 2)then
    (bind ?dinamica f)
else
    (bind ?dinamica p)
)
```

Figura 72: Código que almacena la dinámica de las notas.

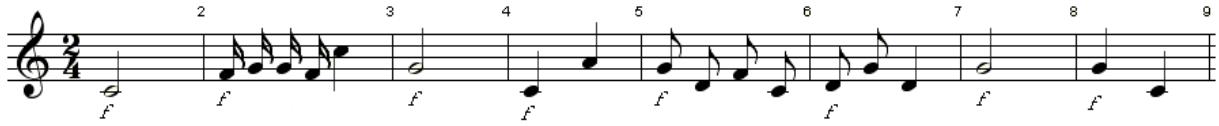


Figura 73: Ejemplo de dictado melódico (III).

En el ejemplo de arriba, Figura 73, se pueden ver las decisiones explicadas anteriormente para la dinámica. Se ha indicado con una *f* aquellas notas que sonarían *forte.* Como se ve, en todos los casos, es la primera nota de cada compás. La dinámica permite al usuario identificar el tiempo del compás, en este caso 2/4, al calcular cuantos tiempos pasan desde que suena *forte* una nota hasta que vuelve a sonar la siguiente nota *forte.*

5.2.2. Funcionamiento de la memoria de trabajo

Cada vez que se ejecute el motor de reglas, se generarán una serie de datos que se guardarán en la memoria de trabajo a través de los campos de las plantillas, explicadas en el Capítulo 3.2.1.1. A continuación se explican las dos plantillas utilizadas para la creación de los dictados.

```
(deftemplate tiempoCompas
    " "
    (slot numerador (type INTEGER))
    (slot denominador (type INTEGER))
    )
```

Figura 74: Plantilla que define el tiempo del compás en Jess

`tiempoCompas` → nombre de la plantilla.

`numerador` → numerador de la fracción que define el tiempo del compás.

`denominador` → denominador de la fracción que define el tiempo del compás.

Esta plantilla sirve para definir el tiempo del compás de todo el dictado. Como se ha explicado en el apartado 2.2, el tiempo del compás de una partitura se mide por medio de un quebrado con dos cifras o fracción de compás. Para poder recuperar posteriormente el tiempo del compás para crear visualmente la partitura, se insertarán los valores de esta fracción en la memoria de trabajo. Únicamente contiene el valor del numerador y del denominador de la fracción que determina el tiempo del compás.

Debido a que el tiempo del compás es el mismo durante todo el dictado, será necesario almacenarlo una única vez en la memoria de trabajo. Es decir, será necesaria una única línea de inserción mediante esta plantilla. Estos datos podrían ser insertados en la memoria de trabajo en cualquier momento de la creación de las notas, ya que será posteriormente, una vez guardados todos los datos, cuando se recuperarán para crear la partitura. Gráficamente, en los pentagramas, la fracción del tiempo del compás se escribe al comienzo del mismo. En este sistema se almacena dicha fracción en primer lugar, antes de almacenar ningún dato sobre las notas.

```
(assert (tiempoCompas (numerador ?*restanteCompas*)
                      (denominador ?*base*)))
```

Figura 75: Línea de inserción del tiempo del compás en la memoria de trabajo.

`assert` → es la orden con la que se insertan datos en la memoria de trabajo.

`tiempoCompas` → nombre de la plantilla en la se que insertan datos.

`numerador y denominador` → son los campos de la plantilla explicados anteriormente.

`?*restanteCompas*` → indica cuantos tiempos le quedan al compás por completar. Su valor es el mismo al principio de cada compás, antes de insertar ninguna nota en el mismo.

`?*base*` → indica el valor de una nota en cada unidad del compás.

Esta segunda plantilla se utiliza para definir diferentes atributos de las notas que forman parte del dictado. Estos atributos han sido explicados en el Capítulo 2.1. Cada vez que se quiera insertar una nota en la memoria de trabajo, es necesario hacerlo según la plantilla *composicion*, explicada anteriormente. A continuación se muestra un ejemplo de una inserción de datos en la memoria de trabajo a través de ésta plantilla:

```
(deftemplate composicion
    " "
    (slot numeroNota)
    (slot forte)
    (slot compas)
    (slot nota (type INTEGER))
    (slot longitud))
```

Figura 76: Plantilla de los atributos de las notas

`composición` → nombre de la plantilla.

`numerosota` → va contabilizando el número de pulsos que se van generando.

`forte` → guardará el valor correspondiente a la dinámica de la nota.

`compas` → guardará el número de compás en que está cada nota.

`nota` → guardará el valor de la frecuencia de la nota mediante el entero del estándar MIDI.

`longitud` → guardará el valor correspondiente a la figura musical.

```
(assert (composicion (numeroNota ?*contador*) (forte ?dinamica)
(compas ?*numCompas*) (nota ?nota) (longitud ?longitud)))
```

Figura 77: Línea de inserción de los atributos de un nota en la memoria de trabajo.

A continuación se va a detallar el significado de cada una de los términos de la línea de inserción en la memoria de trabajo:

`assert` → es la orden con la que se insertan datos en la memoria de trabajo.

`composición` → nombre de la plantilla en la que se insertan datos.

`numeroNota, forte, compás, nota, longitud` → son los campos de la plantilla explicados anteriormente.

`?*contador*` → es un contador que va sumando el valor de las longitudes de las notas.

`?dinamica` → indica la dinámica de la nota, si es forte o piano.

`?*numCompas*` → almacena el número del compás en el que está esa nota.

`?nota` → almacena lo que correspondería a la frecuencia de la nota.

`?longitud` → indica la longitud temporal de la nota.

Para que la inserción de datos sea correcta, cada sentencia de inserción debe ser diferente a todas las anteriores. Para ello se van sumando las longitudes de las notas ya insertadas hasta ese momento en la variable `?*contador*`.

Una vez que se ejecuta el motor de reglas, y se guardan los datos en la memoria de trabajo, se puede mostrar el contenido de esta memoria. La Figura 78 muestra un ejemplo del contenido de la memoria de trabajo.

```
f-0   (MAIN::initial-fact)
f-3   (MAIN::tiempoCompas (numerador 2) (denominador 4))
f-4   (MAIN::composicion (numeroNota 0.5) (forte f) (compas 1) (nota 60) (longitud 0.5))
f-5   (MAIN::composicion (numeroNota 1.0) (forte p) (compas 1) (nota 72) (longitud 0.5))
f-7   (MAIN::composicion (numeroNota 1.25) (forte p) (compas 1) (nota 71) (longitud 0.25))
f-8   (MAIN::composicion (numeroNota 1.5) (forte p) (compas 1) (nota 72) (longitud 0.25))
f-9   (MAIN::composicion (numeroNota 1.75) (forte p) (compas 1) (nota 67) (longitud 0.25))
f-10   (MAIN::composicion (numeroNota 2.0) (forte p) (compas 1) (nota 67) (longitud 0.25))
f-12   (MAIN::composicion (numeroNota 3.0) (forte f) (compas 2) (nota -2147483648) (longitud 1))
f-14   (MAIN::composicion (numeroNota 4.0) (forte p) (compas 2) (nota 64) (longitud 1))
f-16   (MAIN::composicion (numeroNota 5.0) (forte f) (compas 3) (nota 69) (longitud 1))
f-18   (MAIN::composicion (numeroNota 5.25) (forte p) (compas 3) (nota 64) (longitud 0.25))
f-19   (MAIN::composicion (numeroNota 5.5) (forte p) (compas 3) (nota 62) (longitud 0.25))
f-20   (MAIN::composicion (numeroNota 5.75) (forte p) (compas 3) (nota 60) (longitud 0.25))
f-21   (MAIN::composicion (numeroNota 6.0) (forte p) (compas 3) (nota 64) (longitud 0.25))
f-23   (MAIN::composicion (numeroNota 7.0) (forte f) (compas 4) (nota 71) (longitud 1))
f-25   (MAIN::composicion (numeroNota 8.0) (forte p) (compas 4) (nota 67) (longitud 1))
f-27   (MAIN::composicion (numeroNota 8.25) (forte f) (compas 5) (nota 64) (longitud 0.25))
f-28   (MAIN::composicion (numeroNota 8.5) (forte p) (compas 5) (nota 72) (longitud 0.25))
f-29   (MAIN::composicion (numeroNota 8.75) (forte p) (compas 5) (nota 67) (longitud 0.25))
f-30   (MAIN::composicion (numeroNota 9.0) (forte p) (compas 5) (nota 67) (longitud 0.25))
f-32   (MAIN::composicion (numeroNota 9.25) (forte p) (compas 5) (nota 64) (longitud 0.25))
f-33   (MAIN::composicion (numeroNota 9.5) (forte p) (compas 5) (nota 72) (longitud 0.25))
f-34   (MAIN::composicion (numeroNota 9.75) (forte p) (compas 5) (nota 60) (longitud 0.25))
f-35   (MAIN::composicion (numeroNota 10.0) (forte p) (compas 5) (nota 64) (longitud 0.25))
f-37   (MAIN::composicion (numeroNota 12.0) (forte f) (compas 6) (nota 60) (longitud 2))
f-39   (MAIN::composicion (numeroNota 13.0) (forte f) (compas 7) (nota 65) (longitud 1))
f-41   (MAIN::composicion (numeroNota 13.5) (forte p) (compas 7) (nota 65) (longitud 0.5))
f-42   (MAIN::composicion (numeroNota 14.0) (forte p) (compas 7) (nota 60) (longitud 0.5))
f-44   (MAIN::composicion (numeroNota 15.0) (forte f) (compas 8) (nota 67) (longitud 1))
f-45   (MAIN::composicion (numeroNota 16.0) (forte p) (compas 8) (nota 60) (longitud 1))
```

Figura 78: Ejemplo del contenido de la memoria de trabajo.

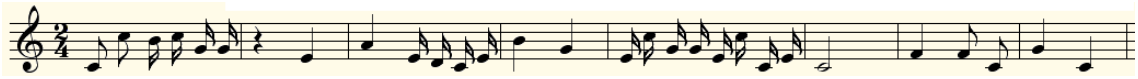


Figura 79: Dictado construido a partir de la memoria de trabajo de la Figura 78

En la Figura 78, se puede ver que en primer lugar se almacena una única vez el tiempo de compás mediante la plantilla `tiempoCompas` y posteriormente treinta notas con diferentes tonos, longitudes y dinámica.

Se puede ver que el tono de la primera nota es 60 (Do4) y los dos últimos 67 y 60 (Sol4 y Do4), lo que correspondería a una tonalidad de Do M donde la tónica es Do y la dominante Sol. Se cumplirían las reglas sobre el tono.

El tiempo del compás es un $\frac{2}{4}$. Si se suman las longitudes de cada compás, se verá que suma siempre 2. Además, las corcheas (longitud 0.5) aparecen siempre de dos en dos y las semicorcheas (longitud 0.25) en grupos de cuatro. Se cumplirían las reglas sobre la longitud.

Además, la primera nota de cada compás tiene dinámica *forte* y el resto *piano,* se cumpliría la regla de la dinámica.

De esta manera, siempre que se necesita guardar información, se puede hacer a través de la memoria de trabajo. Posteriormente se podrá recuperar accediendo de distintas maneras a esta memoria.

El motor de reglas se puede ejecutar desde la consola que proporciona Jess o desde cualquier clase Java. Este sistema ejecuta el motor de reglas desde la clase Java.

5.3. Constructor

El módulo Constructor incluye otros tres módulos, Reestructurador, Integrador y WriteMIDI. Los tres módulos han sido desarrollados mediante el leguaje de programación Java junto con la herramienta jMusic, que proporcionar una biblioteca de composiciones musicales y herramientas de procesamiento de audio, como se ha descrito en el apartado 3.3.1.

5.3.1. Reestructurador

En primer lugar, desde la clase Java se accederá a la memoria de trabajo y se recuperarán los datos mediante el algoritmo Rete, el cual es capaz de almacenar los hechos de la memoria de trabajo en forma de lista iterativa en la que cada entrada es un hecho de la memoria de trabajo Jess.

```
Iterator notasDictado = rete.listFacts();
```

Figura 80: Código que recupera los hechos de la memoria de trabajo y los almacena en una lista iterativa.

Cada entrada de la lista tiene diferentes atributos, cada uno de los almacenados en la memoria de trabajo. La lista iterativa se puede asemejar a una tabla donde cada fila es un hecho y cada columna un atributo. De esta manera se podrá recuperar cada hecho por separado y dentro de éste, la información relativa a cada uno de los atributos.

Tabla 14: Ejemplo de lista iterativa.

Hechos	**Atributos de las notas**				
	numeroNota	forte	compas	nota	longitud
hecho 1	0.5	f	1	60	0.5
hecho 2	1	p	1	67	0.5
hecho 3	2	p	1	69	1
hecho 4	3	f	2	60	1
hecho5	4	p	2	62	1
…					

Una vez que se tiene creada la lista iterativa, se analiza el primer hecho, correspondiente a la plantilla `tiempoCompas` y se toma la información de los distintos campos. Posteriormente, se procede al análisis del resto de hechos, correspondientes a la plantilla `composición`. Para poder recuperar cada campo, éstos se enumeran en el orden en que fueron definidos en la plantilla, mostrada en la Figura 76. La clase Java extraerá la información almacenada únicamente en los campos 0, 1 y 3 de cada hecho o entrada de la lista iterativa.

Tabla 15: Orden de los campos en la plantilla.

Número de orden	Campos de la Plantilla
0	numeroNota
1	Forte
2	Compas
3	Nota
4	Longitud

Cada vez que se recupere un atributo de una nota, la información es almacenada en un vector diferente. Si el atributo es referente al tono de la nota, se almacenará en el vector de tonos de notas. Si es referente a la longitud, será almacenado en su vector correspondiente. Y con la dinámica se actuaría de la misma manera.

Una vez que se ha almacenado la información relativa al primer hecho, se continúa con el siguiente hecho hasta recorrer toda la lista iterativa.

De esta manera, se habrán creado tres vectores, con un tamaño igual al número de símbolos que tenga el dictado. Por lo tanto, el tamaño de los tres vectores será el mismo.

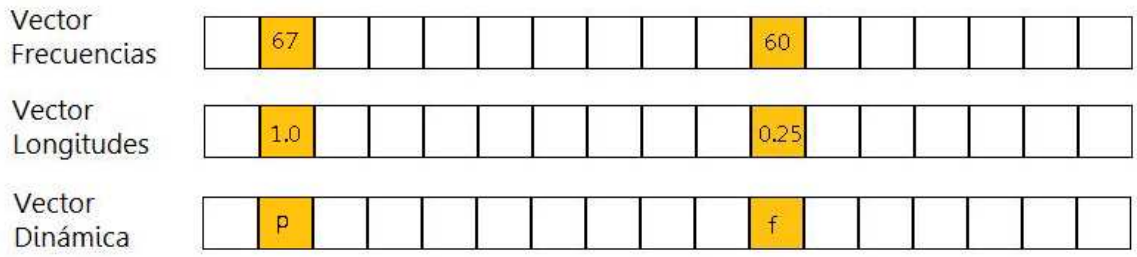


Figura 81: Vectores que almacenan la secuencia de la información necesaria para la construcción de las notas.

```
while(notasDictado.hasNext()){

   Fact f=(Fact)notasDictado.next();

   if(f.getName().equals("MAIN::tiempoCompas")){

      numAux=f.get(0).toString();
      denAux=f.get(1).toString();
      numerador=Integer.valueOf(numAux).intValue();
      denominador=Integer.valueOf(denAux).intValue();
   }
   if(f.getName().equals("MAIN::composicion")){

      factNotas = f.get(3).toString();
      intNotas = Integer.valueOf(factNotas).intValue();
      vect_notas.addElement(intNotas);

      factLongitudes = f.get(4).toString();
      doubleLongitudes= Double.valueOf(factLongitudes).doubleValue();
      vect_longitudes.addElement(doubleLongitudes);

      factDynamic = f.get(1).toString();
      vect_dynamic.addElement(factDynamic);

   }
}
```

Figura 82: Código que extrae la información de la memoria de trabajo y la almacena en vectores.

El vector de *frecuencias*, guarda los valores de las frecuencias de las diferentes notas codificadas según jMusic. A continuación se muestra una tabla de las distintas frecuencias que pueden tener los dictados.

Tabla 16: Relación de notas, valores fijados por jMusic, y frecuencias posibles en un dictado.

Nota	Valor MIDI	Frecuencia (en Hz)
Do4	60	261,63
Do4#	61	277,16
Re	62	293,66
Re#	63	311,13
Mi	64	329,63
Fa	65	349,23
Fa #	66	369,99
Sol	67	392,00
Sol#	68	415,30
La	69	440,00
Si b	70	466,16
Si	71	493,88
Do5	72	523,25
Do5#	73	554,37
Re	74	587,33
Silencio	-2147483648	0

El vector de *longitudes* guarda la duración de cada nota, según la correspondencia de jMusic.

Tabla 17: Figuras posibles en los dictados y sus valores.

Figura	Valores
Blanca	2.0
Negra	1.0
Corchea	0.5
Semicorchea	0.25

El vector de *dinámica* almacena la dinámica de la nota, es decir, su intensidad sonora. La nota puede sonar mezzo*piano (más debil)* o *forte (más fuerte)*, lo que correspondería con las constantes MEZZO_PIANO (valor 60) y FFF (valor 127) respectivamente para jMusic. La decisión de si suena *mezzopiano o forte* se realiza mediante reglas que comprueban si la nota es la primera del compás o no. En el caso de que sea la primera nota tocada en el compás, deberá sonar con más intensidad, es decir *forte*. El resto de notas, deberán sonar con menor intensidad, es decir, *mezzopiano.*

En las Figura 84, Figura 85, y Figura 86 se puede ver un ejemplo de un dictado y sus vectores correspondientes resultados de esta fase.

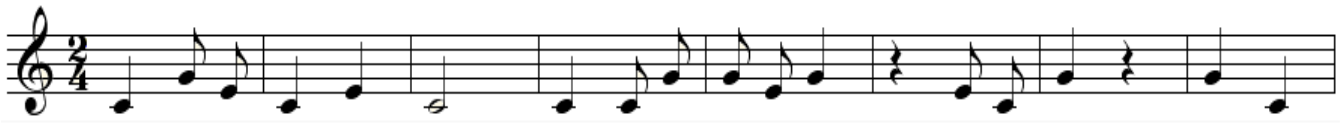


Figura 83: Ejemplo de dictado.

```
[60, 67, 64, 60, 64, 60, 60, 60, 67, 67, 64, 67,
 -2147483648, 64, 60, 67, -2147483648, 67, 60]
```

Figura 84: Vector de frecuencias.

```
[1.0, 0.5, 0.5, 1.0, 1.0, 2.0, 1.0, 0.5, 0.5,
0.5, 0.5, 1.0, 1.0, 0.5, 0.5, 1.0, 1.0, 1.0, 1.0]
```

Figura 85: Vector de longitudes.

```
[f, p, p, f, p, f, f, p, p, f, p, p, f, p, p, f, p, f, p]
```

Figura 86: Vector de dinámica.

5.3.2. Integrador

Una vez se tiene la información extraída de la memoria de trabajo y guardada en los tres vectores, se puede encapsular en pequeñas unidades musicales. Esta unidad es la denominada *Note* por jMusic. *Note* hace referencia no únicamente al tono de la nota, sino a los tres atributos, tono, duración y dinámica. Para poder realizar esta operación se accederá una a una a todas las posiciones de cada uno de los tres vectores, tomando los tres atributos de cada posición y encapsulando los tres datos en un objeto *Note*.

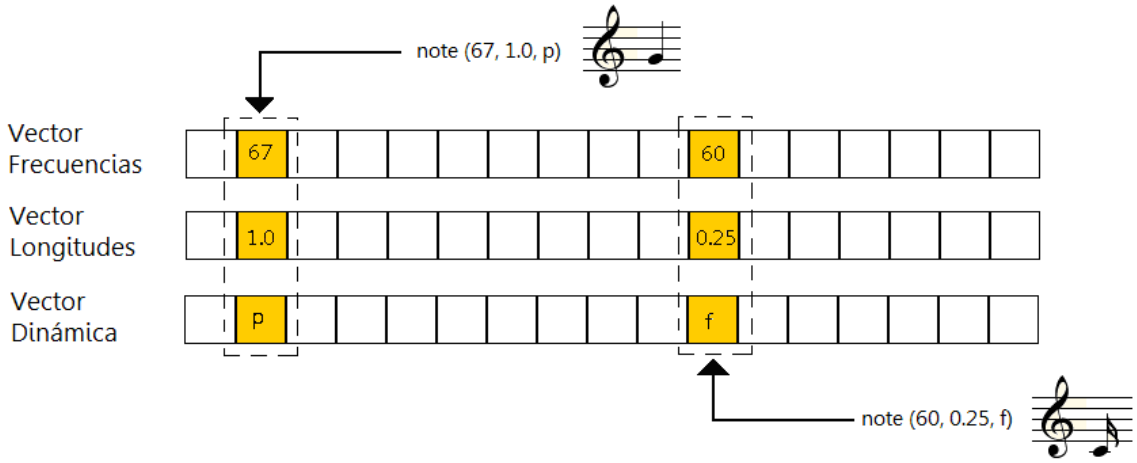


Figura 87: Recuperación de la información relativa a las nota.

Tal como se describe en el apartado 2.2.1, la partitura (Score) se compone de partes (Part), que a su vez se compone de frases (phrase), la cual contiene todas las notas (Note). Cada vez que se construye una unidad Note, ésta se añade a la unidad superior. Esto mismo sucede con el resto de unidades hasta completar la partitura final. Una vez creada la partitura final, se le pueden definir diferentes atributos como el título, el tiempo del compás que va a tener el dictado, etc.

```
while(i<vector_longitudes.size()){

   notaIn = vector_notas.elementAt(i).toString();
   nota = Integer.valueOf(notaIn).intValue();

   durac =(Double)vector_longitudes.elementAt(i);

   dynam = vector_dynamic.elementAt(i).toString();
   note = new Note(nota,durac,dynam);

   frase.addNote(note);

   i++;
}

parte.addPhrase(frase);
score.addPart(parte);
score.setTitle("Dictado");
score.setTimeSignature(numerador,denominador);
```

Figura 88: Código que encapsula la información en las distintas notas y construye la partitura con sus atributos.

Debido a la reducida duración de los dictados al nivel tratado en esta aplicación, la partitura de los dictados contendrá una única parte, ésta a su vez una única frase y será la frase la que contenga todas las notas. El resultado de este módulo es la partitura del dictado.

5.3.3. WriteMidi

Una vez que el sistema ha creado la partitura con la información relativa a las notas, se da lugar a la construcción de un archivo de audio.

jMusic ofrece la posibilidad de construir un archivo de audio a partir de una partitura generada anteriormente. Los formatos con los que permite trabajar son el formato propio de jMusic (.jm), el estándar MIDI, formatos de audio AU, WAVE y AIFF, o archivos en xml.

En este caso se ha elegido MIDI porque es un formato estándar que entienden un gran número de máquinas y reproductores, por lo que se podrá exportar fácilmente.

5.4. Interfaz de Usuario de Salida

Una vez que el sistema ha generado partitura, se presentan al usuario distintas opciones por consola y se le pedirá que introduzca la respuesta por teclado que el sistema leerá e interpretará.

```
Para escuchar el dictado, pulse la tecla E. Para ver el dictado, pulse V:
```

Figura 89: Opciones mostradas al usuario.

Si la tecla pulsada es la "E", el sistema utilizará el reproductor que proporciona jMusic para reproducir la partitura. Si se pulsa la tecla "V", el sistema se servirá de la interfaz que proporciona esta misma biblioteca de Java para la representación gráfica de la partitura.

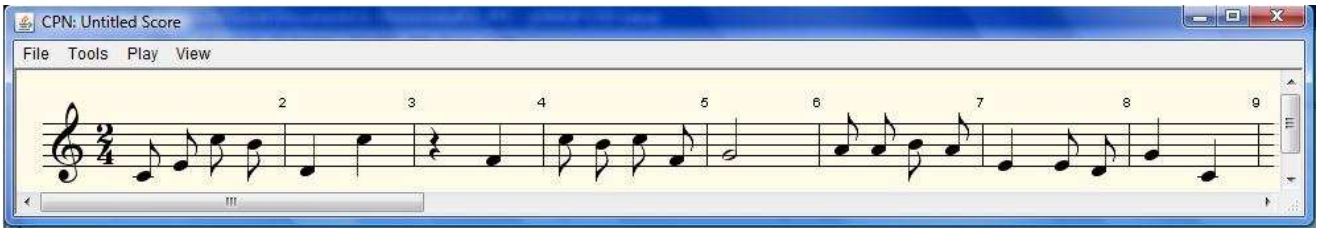


Figura 90: Ejemplo de dictado con la interfaz.

EVALUACIÓN

La evaluación de este sistema se ha realizado desde dos enfoques diferentes. El primero de ellos se refiere al funcionamiento, es decir, se ha comprobado que éste sea correcto. El segundo de ellos es la valoración del sistema por parte de los usuarios.

5.5. Funcionamiento del sistema

Para poder comprobar el buen funcionamiento del sistema, se han generado un número alto de dictados y se ha verificado que realmente contiene las características que debe de tener.

Como se ha explicado anteriormente, existen dictados de dos tipos y para cada uno de ellos, un conjunto de niveles. Para la evaluación del sistema se han generado dos dictados de cada uno de ellos.

Para los dictados rítmicos hay que comprobar que no existen diferentes tonos de la nota, si las figuras rítmicas corresponden con las definidas en el nivel correspondiente según la Tabla 11 y si la dinámica suena en correspondencia con el tiempo del compás asignado al nivel.

Según los resultados obtenidos, mostrados en el anexo A, se puede observar en todos los casos que los dictados se han generado correctamente.

Para los dictados melódicos, hay que comprobar que el rango de notas y el conjunto de figuras rítmicas que aparecen en los dictados es el especificado en cada nivel. Además habrá que comprobar, como en los dictados rítmicos, si la dinámica es la correcta.

En todos lo dictados melódicos obtenidos se ha observado que los resultados son correctos y que se cumplen las propiedades diseñadas para cada uno de ellos.

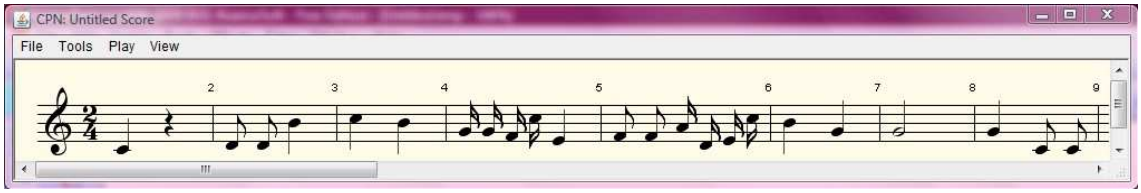


Figura 91. Ejemplo de dictado generado como prueba.

En la Figura 91 se puede observar un ejemplo de los dictados generados como prueba. Corresponde con el Nivel 4 de los dictados melódicos. Según la Tabla 12, el nivel 4 de este tipo de dictados debería tener las características mostradas en la Tabla 18.

Tabla 18. Características del nivel 4 de dictados melódicos.

Nivel 4	Do M	0	2/4	I – VII	

Como se ve, el dictado de la Figura 91 posee las características que se concretaron para ese nivel. Las posibles figuras rítmicas, los posibles tonos de las notas, el tiempo del compás y la tonalidad del dictado entran dentro de los rangos definidos.

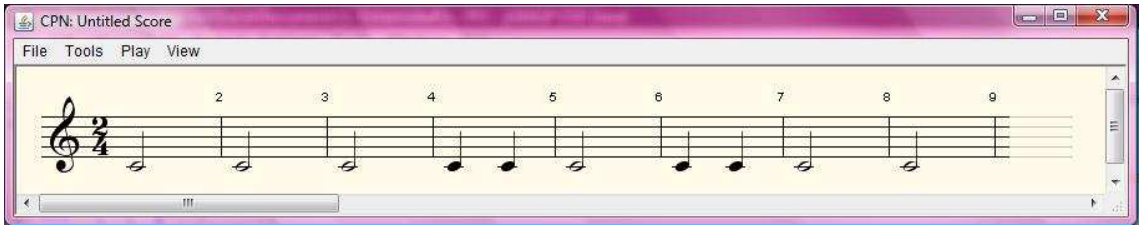
5.5.1. Detalle de los resultados de las pruebas

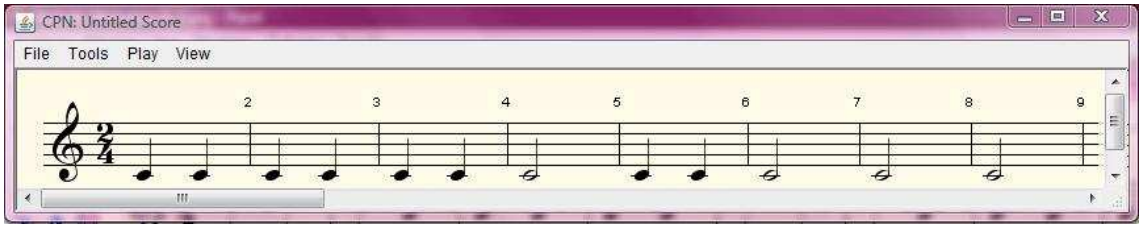
A continuación se muestran los resultados obtenidos con cada prueba referida a cada tipo de dictado y nivel.

A. <u>Tipo de dictado: Rítmico</u>

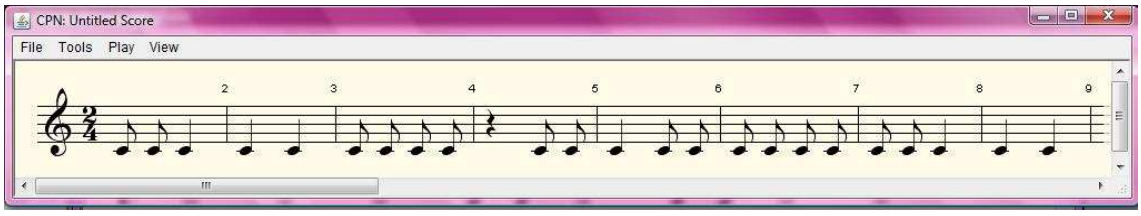
Las características que cada nivel de los dictados rítmicos debería tener están recogidas en la Tabla 11.

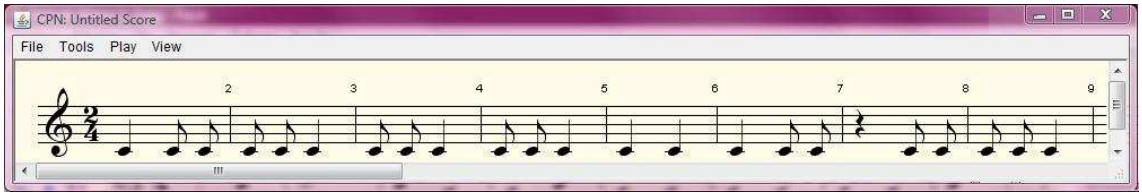
Nivel 0



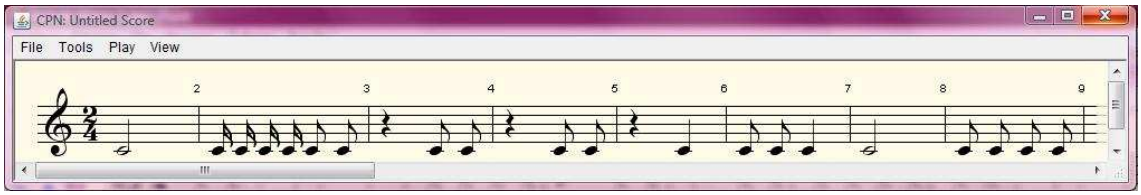


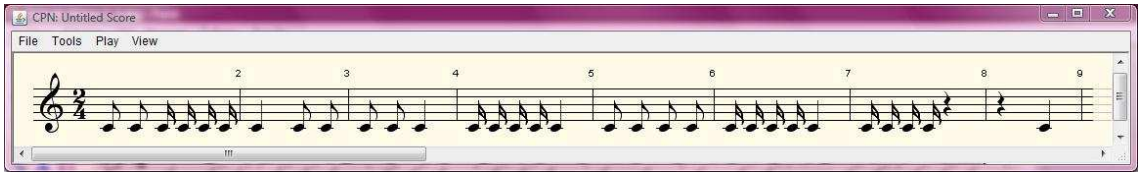
Nivel 1



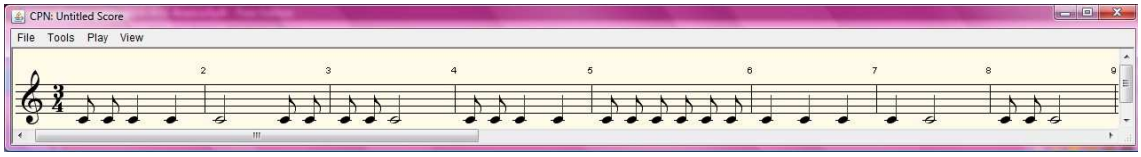


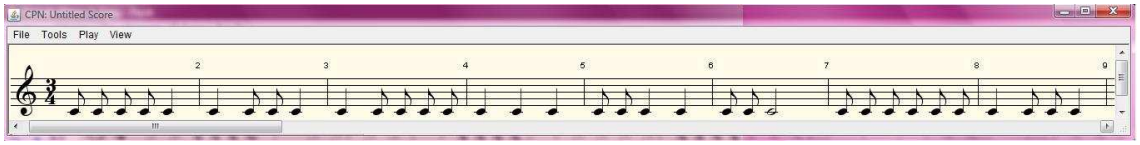
Nivel 2



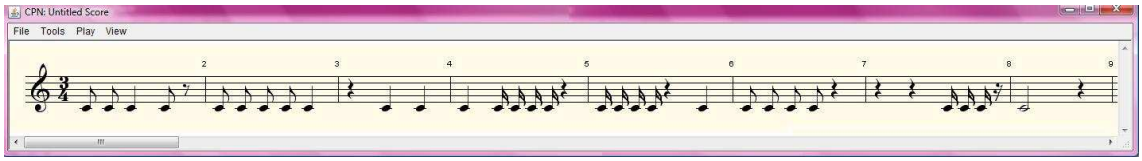


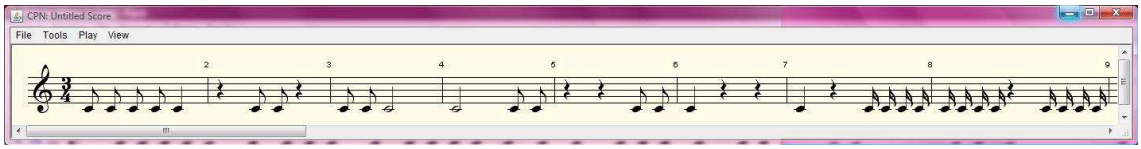
Nivel 3





Nivel 4

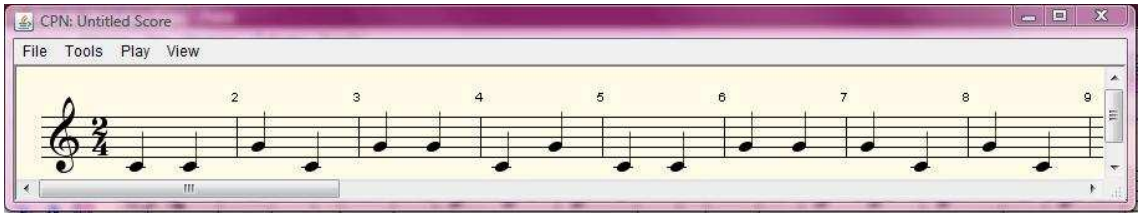


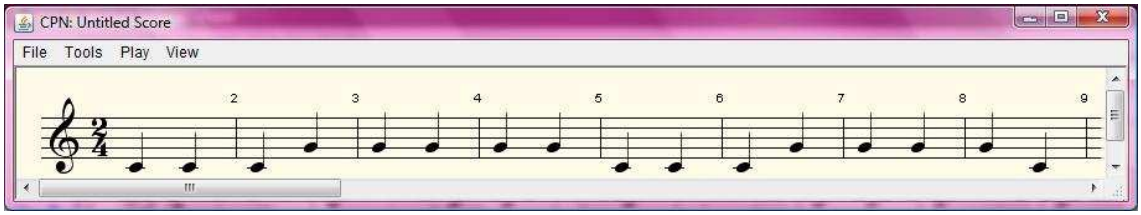


B. Tipo de dictado: Melódico

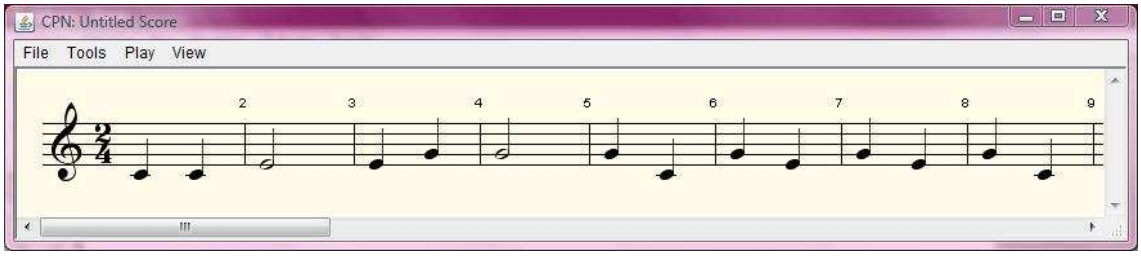
Las características que cada nivel de los dictados rítmicos debería tener están recogidas en la Tabla 12.

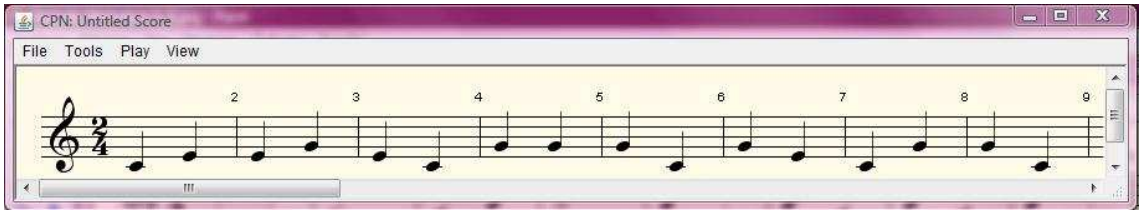
Nivel 0



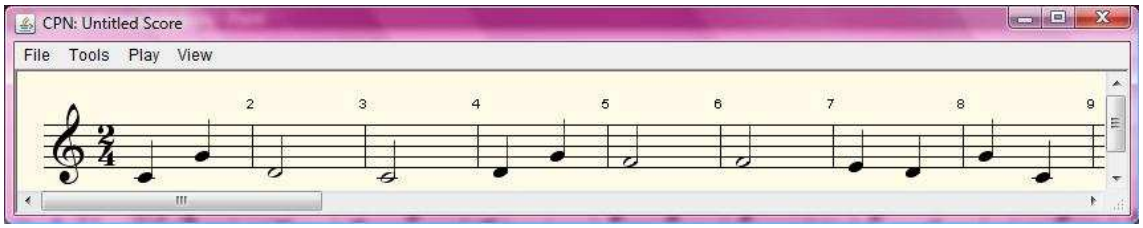


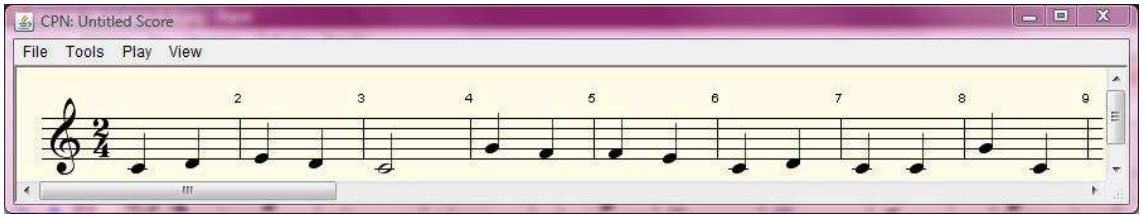
Nivel 1



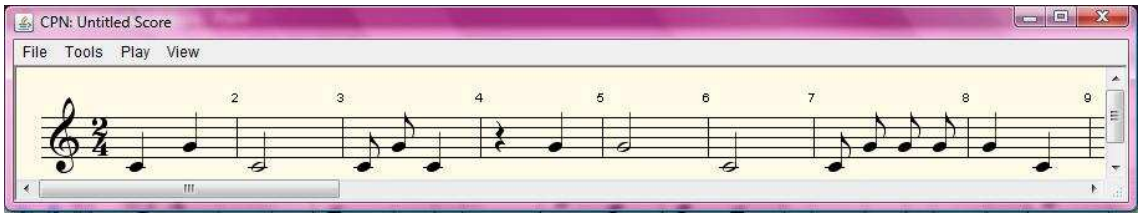


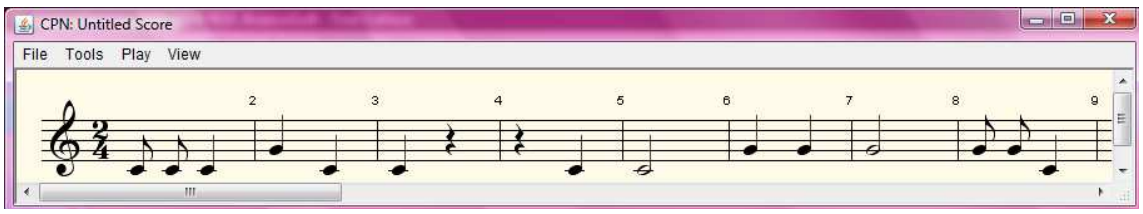
Nivel 2



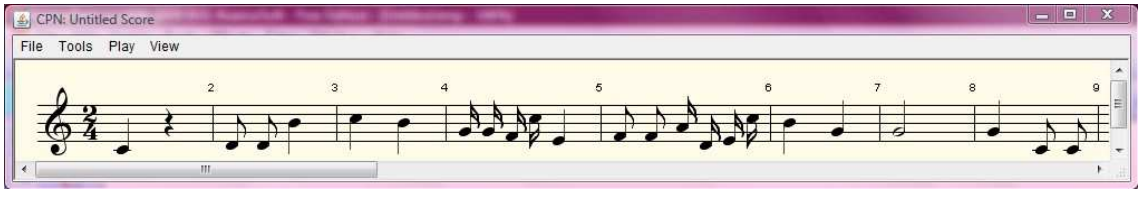


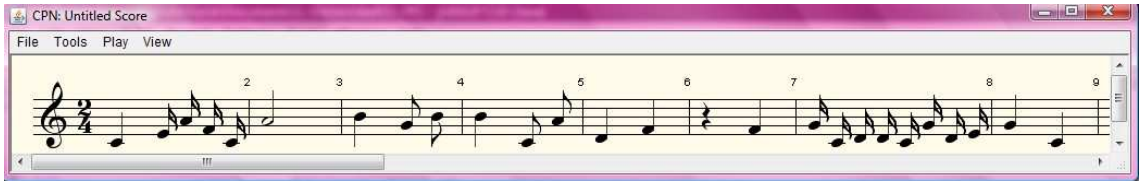
Nivel 3



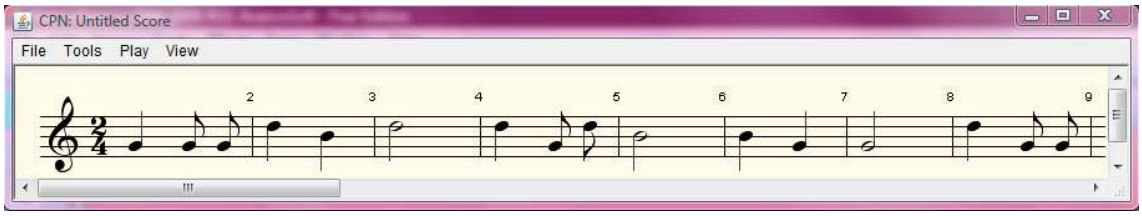


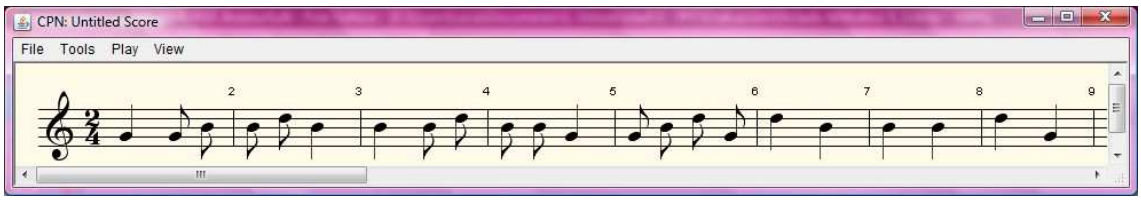
Nivel 4



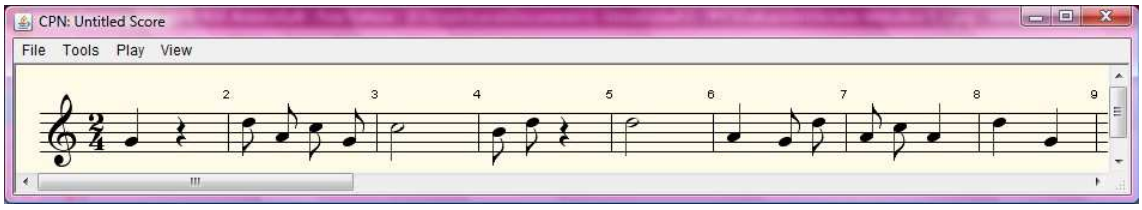


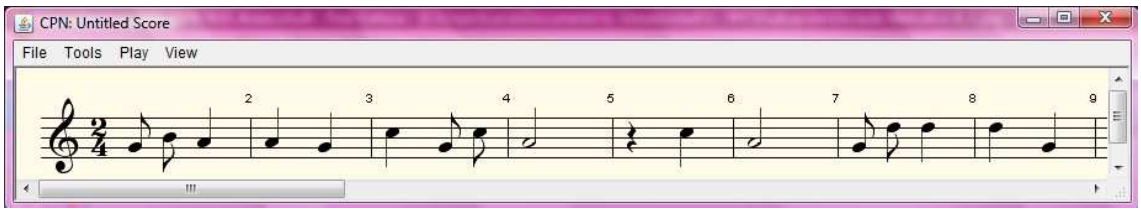
Nivel 5



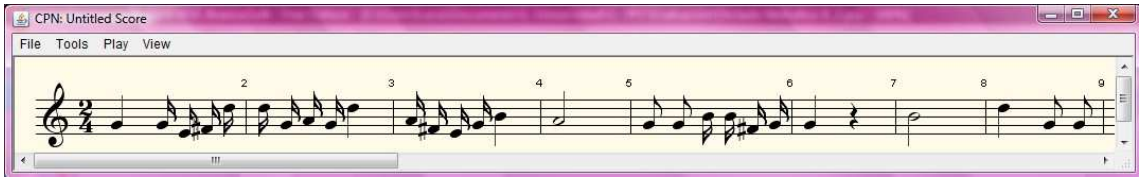


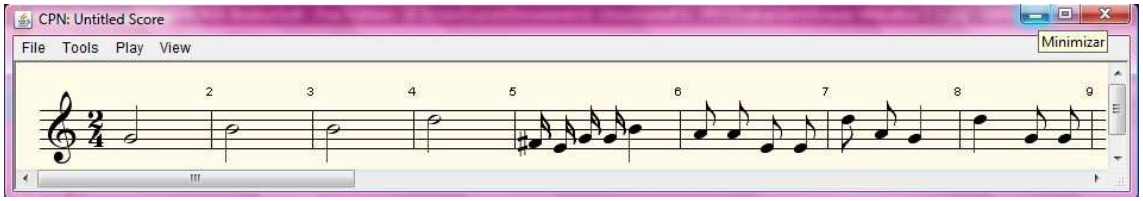
Nivel 6



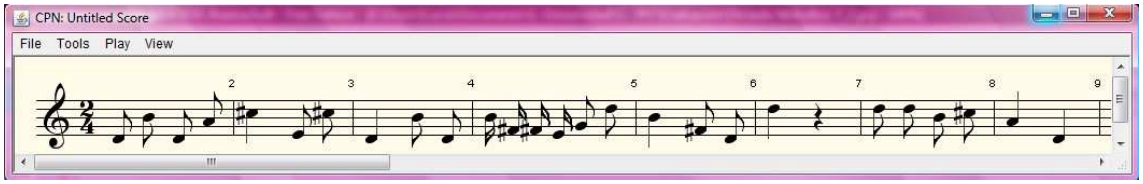


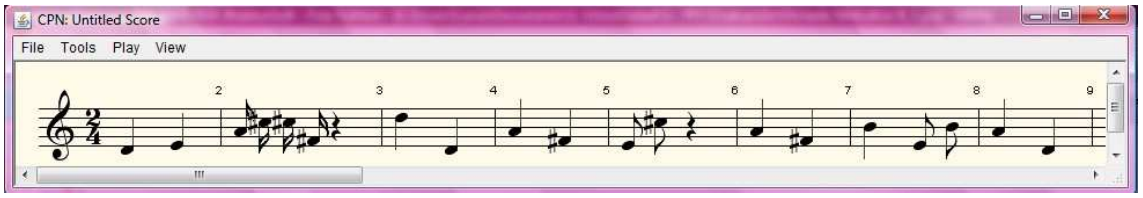
Nivel 7



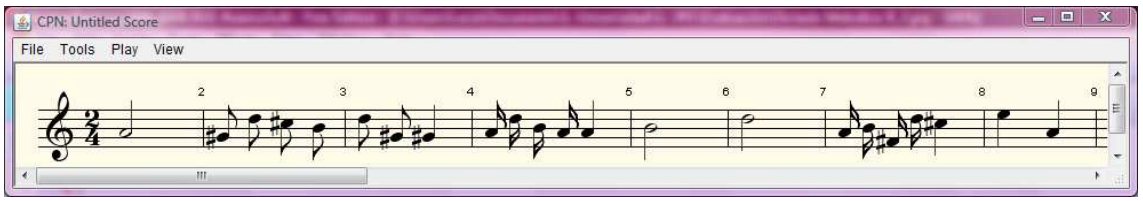


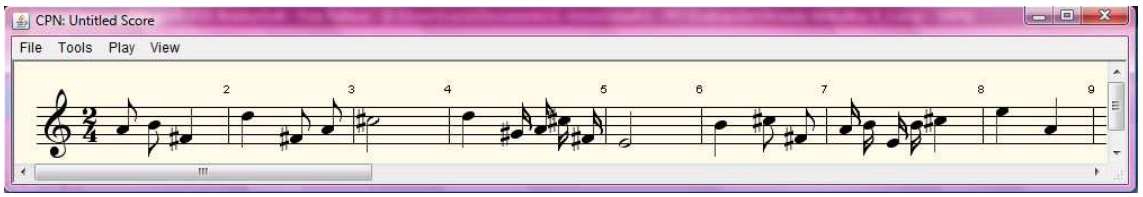
Nivel 8



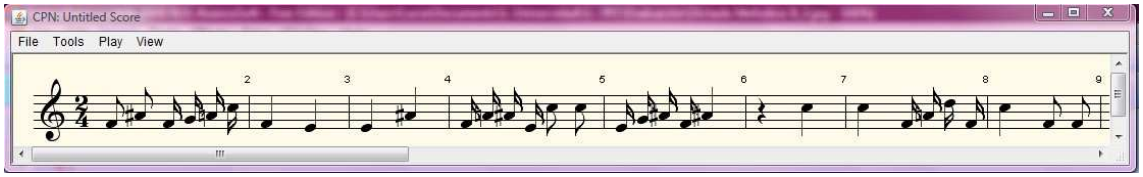


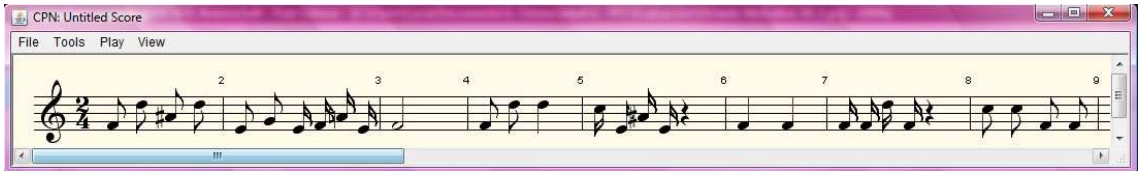
Nivel 9



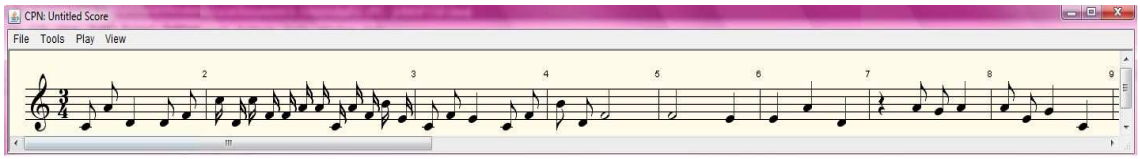


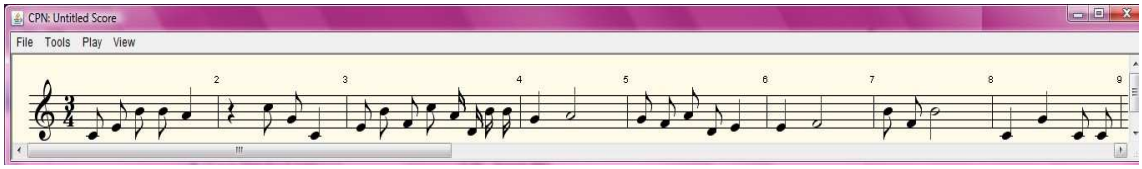
Nivel 10





Nivel 11





5.6. Valoración de los usuarios

Este sistema se ha creado con el objetivo de ofrecer a los usuarios un sistema que solucione un problema que se presenta en los estudios musicales. Por lo que el sistema no sería útil si no lo es para los usuarios.

Es por eso, que se ha ofrecido a 10 posibles usuarios que incluyen alumnos, estudiantes de música, profesores, etc. la posibilidad de evaluar el sistema. Lo más importante a evaluar por los usuarios es la utilidad de un sistema de estas características. Además, también es necesario que se evalúe la calidad de los dictados, si el número de tipos y niveles de dictados es suficiente, qué mejoras se podrían añadir, etc. Para poder evaluar todas estas características, los usuarios han observado el funcionamiento del sistema y han generado una serie de dictados para comprobar los resultados.

A continuación se presenta una síntesis de las valoraciones de los diferentes usuarios, todos ellos con conocimientos de música.

A los usuarios se les ha pedido que valoren de 1 a 5 la utilidad de los dictados, la calidad de los resultados, si piensan que los tipos de dictados ofrecidos satisfacen las necesidades así como el conjunto de niveles. Con todas las valoraciones se ha realizado una media aritmética para cada característica y el resultado se presenta en la Figura 92.

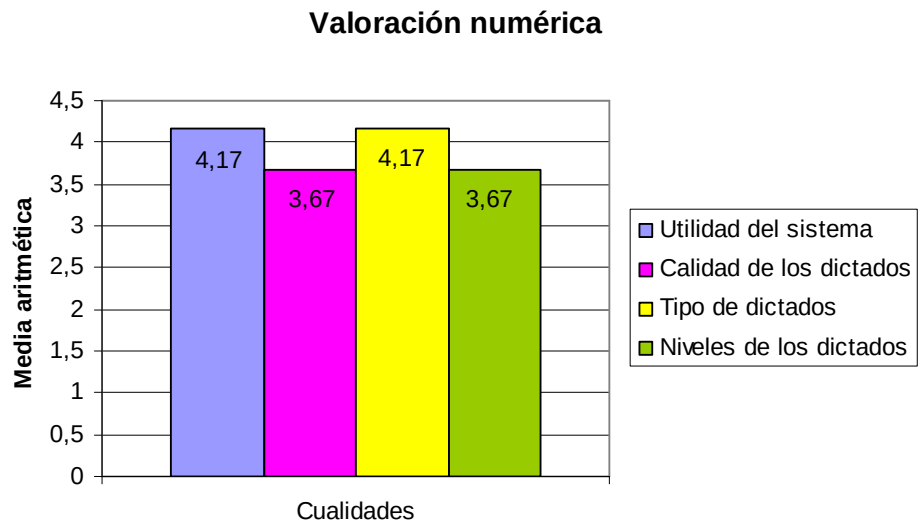


Figura 92. Valoración media de cada una de las características.

Además de la valoración numérica se les ha pedido a los usuarios su opinión sobre una serie de aspectos que podrían mejorar la calidad del sistema.

1.- Qué tipos de dictado añadirían. Entre las propuestas están los dictados de acordes, los dictados de tonos, sin ritmo y los dictados a varias voces.

2.- Qué nuevas características incluirían en los nuevos niveles. Este ha sido uno de los puntos más desarrollados. Entre las propuestas, cabe destacar una mayor variedad en el tiempo del dictado, las figuras (figuras con puntillos, silencios de corchea y semicorchea, ligaduras…), que cada nivel no tuviera un tiempo y una tonalidad concreta sino que pudiera variar.

3.- Puntos positivos del sistema. Entre los puntos positivos del sistema, los entrevistados opinan que éste es un sistema útil, original y fácil de manejar. Además, valoran que cada vez que se genere un dictado, éste sea diferente y se pueda guardar.

4.- Puntos negativos del sistema. Como puntos positivos, la mayoría de los usuarios opinan que la interfaz necesitaría ser más llamativa. Además, piensan que el número de niveles debería ser mayor y que

debería haber más tipos de compases y tonalidades de los dictados deberían ser más.

Todas estas opiniones se han tenido en cuenta a la hora de desarrollar los trabajos futuros ya que son las opiniones de los usuarios las que proporcionan una información más útil sobre las necesidades futuras del sistema.

A continuación se presenta el cuestionario ofrecido a los usuarios para la evaluación del sistema cuyos resultados han sido resumidos en este apartado.

Cuestionario de evaluación

1.- ¿Crees que es útil un sistema de estas características?

1 2 3 4 5

2.- ¿Crees que los dictados tienen la calidad adecuada?

1 2 3 4 5

3.- Respecto al tipo de dictados, ¿crees que son suficientes?

1 2 3 4 5

¿Cuál más añadirías?

4.- Respecto a los niveles, ¿crees que son suficientes?

1 2 3 4 5

¿Qué características más le añadirías a los dictados?

Puntos positivos:
-
-
-

Puntos negativos:
-
-
-

Figura 93: Cuestionario de evaluación.

6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1. Conclusiones

El objetivo del proyecto ha sido desarrollar un sistema inteligente capaz de generar dictados musicales a partir de unas reglas programadas.

El sistema implementado se puede dividir en tres etapas. La primera, la interacción con el usuario tanto para saber cuales son sus preferencias sobre el tipo y el nivel de dictado cómo para mostrarle los resultados. La segunda, la creación del dictado, es decir, la elección de los tonos de las notas, sus longitudes temporales y su dinámica. La tercera, la unificación de los datos en pequeñas unidades, las notas y la posterior construcción de la partitura, la cual se podrá ver o reproducir.

La mayor dificultad de la primera etapa ha sido concretar qué características deberían definir cada nivel. Se ha pensado cuáles serían las necesidades desde el punto de vista del usuario. Además, la interacción con el usuario debe ser sencilla de manera que éste pueda expresar sus preferencias fácilmente.

Para la creación del dictado, realizado en la segunda etapa, se ha utilizado un motor de reglas que ha permitido que las características del dictado sean elegidas en función de unas reglas definidas anteriormente. Estas reglas permiten en cada momento que se elija un tono, una duración y una dinámica determinada, dependiendo de la posición de la nota en el dictado, y del tipo de dictado y nivel elegidos por el usuario en la primera etapa.

Esta segunda etapa es en la que se crean los dictados y por ello es la que se ha desarrollado poniendo más atención a los pequeños detalles. Lo más complicado ha sido establecer las reglas de tal manera que el dictado que se creara no tuviera características indeseadas en su tipo y nivel.

Para la unificación de los datos en las notas, las características escogidas por el sistema experto en la segunda etapa son almacenadas por

separado en función del tipo de característica y en el orden en que fueron creadas. Una a una, se crea cada nota encapsulando en esta unidad las tres características que la definen.

La segunda y tercera etapas han sido desarrolladas mediante diferentes herramientas, Jess y jMusic. A pesar de que ambas están relacionadas con un mismo lenguaje de programación, Java, la interacción de estas dos herramientas diferentes ha sido lo más difícil de conseguir.

Finalmente se crea un archivo de audio que el usuario podrá conservar para su posterior utilización. Además, se ofrecerá la posibilidad al usuario de escuchar o ver el dictado mediante las herramientas proporcionadas por el sistema.

Esto facilita el tener que desarrollar aplicaciones propias que realicen estas funciones. Pero utilizar algo ya creado implica tener que adaptarse a sus características sin poder modificarlas. Esto impide que algunas de las características de los dictados se visualicen de manera diferente a lo musicalmente establecido.

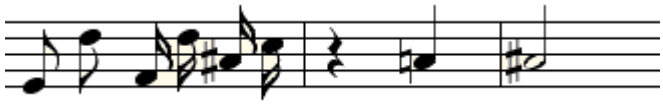


Figura 94. Ejemplo de problemas en las alteraciones.

La Figura 94 muestra un fragmento de la partitura de un dictado melódico de nivel 9. Este nivel, debido a que genera dictados en la tonalidad de Fa M, tiene una alteración en sus notas, el Si b. En la figura se puede observar que en vez de mostrar un Si b, en todo momento muestra La #. El sonido es exactamente el mismo, sin embargo, a efectos de tonalidad, ambos tonos tienen connotaciones diferentes.

La implementación de este sistema mediante un sistema experto implica una serie de ventajas que de otras maneras no existirían. Todos los dictados que genere el sistema tendrán unas características comunes. Además, todos los dictados que sean del mismo tipo y nivel, tendrán otro

conjunto de características propias. Pero ninguno de los dictados es igual a cualquier otro. Aunque es posible que en los niveles inferiores, debido a que las posibilidades de variación no son muy elevadas, los dictados sean más o menos parecidos.

El objetivo del usuario es entrenar su oído y practicar la identificación de las notas, los tiempos, los ritmos, etc. correctamente. Este sistema le permitiría poder generar un dictado diferente cada vez, de manera que el usuario tendría mayor variedad de posibilidades para conseguir su objetivo de aprendizaje sin necesitar un profesor o asistente de estudios. Además, podrá almacenar los dictados para escucharlos de nuevo posteriormente.

La decisión de que los archivos de audio sean en formato MIDI, hace que el usuario los pueda reproducir en cualquier reproductor. La música creada mediante este estándar, es música sintetizada por ordenador. Muchas veces esta música es calificada de menos natural que las grabaciones directas de instrumentos, pero debido a que las melodías de los dictados son sencillas, el sonido de éstos es adecuado para alcanzar el objetivo buscado. Una ventaja frente a las grabaciones directas, es la comentada anteriormente. Cada vez que el sistema genere un dictado, será un dictado diferente. Además, el tamaño de los archivos MIDI es menor que otro tipo de archivos de audio.

6.2. Trabajos futuros

El campo de la música es muy amplio, y debido al gran desarrollo de las tecnologías, las tareas en las que la informática puede ayudar a las personas dedicadas a la música, son muchas.

Los tipos de dictados y los niveles tratados en este sistema son algunos de los que los usuarios necesitarán para su aprendizaje musical.

Una primera línea de actuación sobre este sistema sería la ampliación de estas características, de manera que el usuario pudiera practicar un mayor número de dictados de características diferentes.

Como se ha explicado en el Capítulo 4, nuestro sistema tiene definidos unos niveles, cada uno de ellos con unos parámetros definidos y fijados previamente. Esto hace que para cada nivel, los parámetros nunca cambien y por lo tanto, nunca podrán ser deducidos por el usuario.

Una segunda línea de actuación, relacionada con la anterior, sería hacer los distintos niveles más flexibles. Al igual que en el sistema actual, los niveles irían variando de menor a mayor dificultad, pero cada nivel generaría dictados de diferentes tonalidades, tiempos del compás, etc. El usuario no sabría exactamente cuáles serían estos valores, lo que debería también averiguar, junto con el resto de características.

Este sistema va dirigido a estudiantes que comiencen sus estudios de música. El rango de edad de éstos puede ser amplio y sus conocimientos sobre informática, variados. Es por eso, que es necesario un sistema sencillo y adaptado a todas estas necesidades.

Una tercera línea de actuación más necesario de cara al usuario, es el desarrollo de una interfaz gráfica que permita a los usuarios poder manejar el sistema de manera más sencilla y accesible. Se trataría de una interfaz que mostrara las mismas opciones que el actual sistema, pero añadiendo gráficos más llamativos y sencillos, además de un corrector del dictado.

Una cuarta línea de actuación sería la posibilidad de que el interfaz explicado anteriormente fuera una aplicación Web que permitiera la generación del dictado, su reproducción y visualización a través de esta arquitectura.

Además del generador de dictados, hay otras muchas tareas en las que la informática puede ayudar al estudiante. Un trabajo a realizar en un futuro, puede ser, la ampliación de funcionalidades al sistema. Algunas de estas funcionalidades pueden ser:

- Una interfaz que permita la escritura de partituras propias. De esta manera el alumno podría insertar los diferentes símbolos sobre un pentagrama y así generar sus propias partituras. Además, en el caso

de que el usuario lo solicitara, se podría generar el archivo de audio de esa partitura.

- Armonizador de partituras. Dada una melodía creada digitalmente, por ejemplo con la interfaz explicada anteriormente, el sistema crearía las distintas voces Tenor, Contralto y Bajo.
- Analizador de partituras. Dada una obra musical con sus distintas voces creada digitalmente, por ejemplo con la interfaz explicada anteriormente, se analizaría musicalmente la partitura. Se definiría el tono en el que está la partitura, a qué grado de la escala utilizada correspondería cada nota o acorde, etc.

Referencias

Las siguientes referencias se incluyen por orden de aparición en el texto.

[1] Richard Middleton, *Studying Popular Music*. Philadelphia: Open University Press 1990.

[2] Mario Raja. *Teoría Musical.*[Visitado el 23/08/2009] http://www.aprende-gratis.com/teoria-musical

[3] Diccionario de conceptos musicales.[Visitado el 25/08/2009] http://www.hagaselamusica.com/diccionario/p/

[4] DOBRIAN, C. "Music and artificial antelligence" (1993)[Visitado el 28/08/2009] http://music.arts.uci.edu/dobrian/CD.music.ai.htm

[5] Efecto Mozart. El sonado efecto Mozart. Elena Sanz.)[Visitado el 20/08/2009] http://www.cienciadigital.es/hemeroteca/reportaje.php?id=61

[6] E. Castillo y E. Alvarez. *Sistemas Expertos.* Ed. Paraninfo, S.A., Madrid, 1989.

[7] Alice Agogino. *Introduction to expert systems.* http://best.me.berkeley.edu/~aagogino/me290m/s99/Week2/week2.html

[8] Ed. Patrick Winston and Karen A. Prendergast. *The AI Business: The commercial uses of artificial intelligence*. Cambridge, Mass. (1986)

[9] Clips. A Tool for Building Expert Systems. [Visitado el 30/08/2009] http://clipsrules.sourceforge.net/

[10] SWI-Prolog's**.** [Visitado el 30/08/2009] http://www.swi-prolog.org/

[11] Jess, the Rule Engine for the Java Platform.[Visitado el 30/08/2009] www.jessrules.com

[12] The Rete Algorithm. Jess Manual.[Visitado el 30/08/2009] http://www.jessrules.com/jess/docs/71/rete.html#

[13] Java Sound API.[Visitado el 28/08/2009] http://www.j2ee.me/j2se/1.5.0/docs/api/

[14] JMF Tratamiento multimedia en Java con JMF. Carlos Prades del Valle. Versión 1.0.2. Febrero de 2001.

[15] jMusic. Music composition in Java. [Visitado el 25/08/2009] http://jmusic.ci.qut.edu.au/

[16] Proakis, J. G. y Manolakis, D. G. *Tratamiento digital de señales. Principios, algoritmos y aplicaciones*. Prentice Hall International (UK) Ltd. (1998)

[17] Xavier Blanco. *El protocolo MIDI.* http://www.hispasonic.com/revista/protocolo-midi

[18] MIDI Manufacturers Association [Visitado el 31/09/2009] http://www.midi.org/