



ugr

Universidad
de Granada

TRABAJO FIN DE GRADO
INGENIERÍA INFORMÁTICA

Desarrollo de un Sistema Experto de Análisis Musical

Autor

Laura Olga Tirado López

Directores

Juan Luis Castro Peña



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, Junio de 2017

Desarrollo de un Sistema Experto de Análisis Musical

Laura Olga Tirado López

Palabras clave: sistema experto, sistema web, inteligencia artificial, python, django, clips, pylips, análisis, analizar, música, musical, coro, coral, quintas, octavas, armonía, armónico, melodía, melódico, contrapunto, voces, partitura

Resumen

Se pretende desarrollar una aplicación basada en reglas para poder analizar *corales* o partituras de coro.

El objetivo es implementar un sistema experto que permita a los usuarios analizar partituras de coro a cuatro voces para su posterior corrección, pudiendo así mejorarlas tanto a nivel armónico como a nivel melódico y de contrapunto.

El sistema se basará en conocimiento experto sobre lenguaje musical, composición y armonía clásica proporcionado por varios profesores de armonía y composición del Conservatorio Profesional Ángel Barrios de Granada, músicos titulados y otras fuentes tales como libros y tratados.

Las funcionalidades que proporcionará este sistema se dividirán en dos clases:

- **Análisis armónico:** consistente en la búsqueda de errores que comprendan, al menos, dos de las voces, tales como acordes incompletos, secuencias armónicas incorrectas y búsqueda de disonancias.
- **Análisis melódico:** consistente en la búsqueda de errores referidos a las melodías de las voces, tales como conducción incorrecta de las mismas, falta de coherencia y saltos o movimientos disonantes.

El interés principal de este proyecto es poder desarrollar una herramienta para compositores o estudiantes de composición que haga una función análoga a la de un corrector ortográfico y gramatical de un editor de textos, aplicada a la composición y edición de partituras.

Development of an Expert System for Musical Analysis

Laura Olga Tirado López

Keywords: expert system, artificial intelligence, web system, python, django, clips, pylips, analysis, analyze, music, musical, choir, fifths, eights, harmony, harmonic, melody, melodic, voices, sheet, counterpoint

Abstract

My intention is creating an app that implements an expert system to analyze *choir* sheets.

This project aims to create a rule base engine that allows users analyze choir music sheets for its later correction, being able to improve their harmonies, melodies and counterpoint.

The system will be based on expert knowledge about music notation, composing and classic harmony provided by many harmony and composition teachers from Conservatorio Profesional Ángel Barrios, musicians and other sources such as books and musical tratises.

Features provided by this system will be of two grades:

- **Harmonic analysis:** consisting on search for errors involving, at least, two of the voices, such as incomplete chords, incorrect harmonic sequences and dissonances.
- **Melodic analysis:** consisting on search for errors involving melodies, such as incorrect voice guidance, lack of coherence and dissonant movements.

The main interest of this project is being able to develop a tool for composers or music students that performs a function analogous to an spelling and grammar checker, applied to composing and writing music.

D. **Juan Luis Castro Peña**, Profesor del Área de Inteligencia Computacional del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informa:

Que el presente trabajo, titulado ***Desarrollo de un Sistema Experto de Análisis Musical***, ha sido realizado bajo su supervisión por **Laura Olga Tirado López**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 2017.

El director:

Juan Luis Castro Peña

Agradecimientos

A mi familia, por apoyarme y hacer posible que esté donde estoy hoy.

A mis amigos, por estar siempre ahí y animarme a seguir cuando las cosas se tuercen.

A mis compañeros de informática, porque sin ellos esta etapa que es la vida universitaria no habría tenido tantas risas.

A mis compañeros de la DEIIT, siempre dispuestos a escuchar mis quejas y ayudarme con mis problemas.

A mis expertos, Alberto e Isa, por ayudarme a crear este sistema.

A Jesús, porque el resultado final de este proyecto no hubiera sido el mismo sin ti.

Índice general

Introducción	1
1. Planificación	3
1.1. Fases	3
1.2. Lista de tareas	4
1.3. Recursos humanos	5
1.4. Temporización	6
2. Análisis	8
2.1. Objetivos	8
2.2. Análisis de requisitos	8
2.2.1. Actores	9
2.2.2. Requisitos de información	9
2.2.3. Requisitos funcionales	10
2.2.4. Requisitos no funcionales	16
2.3. Modelos de casos de uso	17
2.3.1. Descripción básica de los actores	17
2.3.2. Descripción de casos de uso	18
2.4. Diagramas de actividad	20
3. Adquisición de conocimiento	21
3.1. Extracción de conocimientos	21
3.2. Educción de conocimientos	22
3.2.1. Incidentes críticos	22
3.3. Estructuración de conocimientos	23
3.3.1. Hechos	23
3.3.2. Reglas	26
3.3.3. Estructura	28
4. Diseño	32
4.1. Selección de herramientas	32
4.1.1. Sistema experto	32
4.1.2. Interfaz	32
4.2. Arquitectura del sistema	35
5. Implementación	37
5.1. Sistema experto	37

5.1.1.	Formato de los datos de entrada: partitura	37
5.1.2.	Implementación del primer prototipo	41
5.1.3.	Modularización del sistema	41
5.2.	Aplicación web	42
5.2.1.	Tratamiento del archivo XML	42
5.2.2.	Integración del sistema experto	43
5.2.3.	Visualización de errores	45
5.3.	Aspecto final	46
6.	Pruebas y validación	49
6.1.	Verificación	49
6.2.	Validación	50
7.	Resultados y conclusiones	51
7.1.	Resultados	51
7.2.	Conclusiones	51
7.3.	Futuro desarrollo	52
	Anexo: Glosario	54
	Bibliografía	59

Índice de figuras

1.4.1.Diagrama de estructura de descomposición de trabajo	6
1.4.2.TempORIZACIÓN de las tareas	7
1.4.3.Diagrama de Gantt	7
2.2.1.Ejemplo de quintas paralelas	11
2.2.2.Ejemplo de quintas directas	11
2.2.3.Ejemplo de octavas paralelas	11
2.2.4.Ejemplo de octavas directas	12
2.2.5.Ejemplo de cuarta aumentada	12
2.2.6.Ejemplo de duplicación de sensible	12
2.2.7.Diagrama de lógica tonal	13
2.2.8.Ejemplo sensible sin resolver	14
2.2.9.Ejemplo de séptima de dominante sin resolver	14
2.2.10Ejemplo de segunda aumentada melódica	14
2.2.11Ejemplo de cuarta aumentada melódica	15
2.4.1.Diagrama de actividad del caso de uso CU-1. Analizar partitura. . . .	20
3.3.1.Diagrama de la estructura del sistema experto	31
4.1.1.Wireframe de la página principal de la interfaz	33
4.1.2.Wireframe de la página de resultados de la interfaz	34
4.1.3.Paleta de colores de la interfaz	34
4.2.1.Diagrama de la arquitectura del sistema	36
5.1.1.Subdivisión de un compás del coral “Aus meines Herzens Grunde” de J.S.Bach a semicorcheas	39
5.3.1.Página principal	46
5.3.2.Página principal tras enviar el formulario	47
5.3.3.Visualización de resultados con errores	47
5.3.4.Visualización de resultados sin errores	48

Índice de tablas

2.3.1.Descripción del actor Ingeniero del Conocimiento	17
2.3.2.Descripción del actor Experto	17
2.3.3.Descripción del actor Usuario	18
2.3.4.Descripción del caso de uso CU-1	18
2.3.5.Descripción del curso normal del caso de uso CU-1	19
2.3.6.Descripción de los cursos alternos del caso de uso CU-1	19

Introducción

La música es un arte que ha acompañado al ser humano desde casi sus inicios. Ésta se ha ido desarrollando a lo largo de los siglos, adaptándose a la época y sus avances, creando nuevos instrumentos y nuevos estilos. Con la evolución de los distintos estilos musicales se fue desarrollando de igual manera la notación musical, desde las primeras notaciones creadas por la cultura griega, a base de letras del alfabeto y algunos símbolos como puntos, hasta la notación abstracta utilizada hoy en día.

En la actualidad, gracias a las nuevas tecnologías, disponemos de editores de partituras que realizan la misma función que los editores de texto que encontramos en cualquier ordenador: poder crear y editar una partitura. Sin embargo, una de las funcionalidades que ofrecen los editores de texto y también los móviles son los correctores ortográficos y/o gramaticales, los cuales en los dispositivos como smartphones y tablets son ampliamente utilizados.

Algunos editores de partituras, como *Sibelius*, en sus últimas versiones sí añaden parte de esta funcionalidad, pudiendo buscar algunos tipos de errores en partituras, tales como quintas u octavas directas. Sin embargo, las reglas de armonía clásica en las que se basan la mayoría de composiciones, salvo las basadas en nuevas corrientes musicales como música contemporánea o experimental, exponen una gran cantidad de restricciones sobre cómo debe componerse una pieza musical de estilo clásico para que sea correcta y coherente. Estas reglas tratan desde la correcta disposición de las notas de un acorde hasta la conducción de melodías y la búsqueda de sonoridades consonantes, evitando las disonancias o utilizándolas de maneras determinadas para conseguir efectos sonoros específicos.

El interés de este proyecto radica pues en la falta de una herramienta de este tipo que permita a compositores y estudiantes de composición y armonía poder corregir errores en sus composiciones o ejercicios de forma automática, con el objetivo de mejorarlas.

Dado que la música abarca una gran variedad de estilos y formas musicales, he decidido para este proyecto centrarme en una de las más extendidas y utilizadas dado su interés para la enseñanza de este campo: los corales o partituras de coro. Esta forma musical proporciona el entorno perfecto para poder estudiar las reglas armónicas, melódicas y de contrapunto al tener cuatro melodías cantadas por cuatro voces independientes que se van entrelazando a lo largo de la obra.

La realización de este proyecto se centra en la resolución de un problema muy con-

creto con un entorno muy específico. Además, aunque hay reglas de armonía que son completamente indiscutibles, existen gran cantidad de excepciones a éstas y cierta subjetividad; algo puede no estar necesariamente mal pero no llegar a ser del todo correcto. Un caso sería la conducción melódica de las voces, la cual puede hacerse de diversas maneras según lo que quiera transmitir el compositor en un momento determinado.

Por estos motivos, mi propuesta para poder implementar esta funcionalidad se basa en un sistema experto, el cual a partir de una base de conocimiento de teoría musical y reglas armónicas pueda analizar las partituras de igual manera que un experto en esta materia, teniendo en cuenta la posible subjetividad del análisis.

En los siguientes capítulos se explicará detalladamente el proceso de desarrollo de este sistema, así como los resultados obtenidos tras su implementación. Además, para facilitar la comprensión de este trabajo, los términos y definiciones musicales que aparecen a lo largo del documento pueden ser consultados en el **Anexo**.

1. Planificación

1.1. Fases

Las fases seguidas a lo largo del desarrollo de este proyecto son las que se detallan a continuación:

- **Fase 1:** Especificación del proyecto. Se establecen los objetivos a cumplir para que el proyecto se considere completado con éxito, entendiendo que se satisfacen todos los requisitos y funcionalidades.
- **Fase 2:** Planificación. Se definen las fases de desarrollo del sistema y las actividades a desarrollar en cada una de ellas.
- **Fase 3:** Análisis. Se realiza el análisis de requisitos del sistema.
- **Fase 4:** Adquisición del conocimiento. Se obtienen los conocimientos necesarios para poder crear el sistema experto.
- **Fase 5:** Diseño. Se analizan todos los aspectos del proyecto para concretar su desarrollo.
- **Fase 6:** Implementación. Se procede a implementar todas las funcionalidades necesarias.
- **Fase 7:** Pruebas y validación. Se verifica y valida el sistema para comprobar su correcto funcionamiento.
- **Fase 8:** Documentación. Se realiza toda la documentación informativa y explicativa.

Aunque parece que estas fases se llevan a cabo de forma lineal, esto no es completamente cierto. La fase de adquisición del conocimiento se lleva a cabo a lo largo de todo el proceso de desarrollo de sistema. Esto se debe al hecho de que al necesitar conocimientos expertos específicos y externos, no conocidos por el desarrollador, es necesario realizar este proceso de adquisición paralelamente al resto de tareas para poder construir de forma correcta el sistema, de acuerdo al ámbito de conocimiento del mismo.

Esto hace que el proceso sea más complejo al tener que hacer una revisión, adquisición y comprensión continua de conocimientos ajenos a la ingeniería informática.

1.2. Lista de tareas

■ Especificación del proyecto:

- Determinación de objetivos. Se definen los objetivos que se quieren alcanzar con el desarrollo de este proyecto.
- Determinación de requisitos. Se definen los requisitos que se deben satisfacer para alcanzar los objetivos definidos.

■ Planificación:

- Lista de actividades. Se especifican las actividades y tareas que deben llevarse a cabo en cada fase del desarrollo.
- Recursos humanos. Se especifica el personal que va a participar en el desarrollo del proyecto y sus funciones.
- Temporización. Se determinan los plazos y fechas para la ejecución de cada una de las fases del proyecto.

■ Análisis:

- Análisis de requisitos. Se definen los requisitos de información, funcionales y no funcionales del sistema.
- Descripción de casos de uso. Se describen los casos de uso del sistema.
- Diagramas de casos de uso.
- Diagramas de actividad.

■ Adquisición del conocimiento:

- Extracción de conocimientos. Se especifica el proceso de adquisición de conocimientos de diversas fuentes no humanas.
- Educción de conocimientos. Se especifica el proceso de adquisición de conocimientos de diversos expertos.
- Estructuración de conocimientos. Se especifican los hechos y reglas que componen el sistema.

■ Diseño:

- Selección de herramientas. Se describe el proceso de elección de herramientas para el desarrollo, tanto del sistema experto como de su interfaz.
 - Diseño de la arquitectura. Se definen los componentes y módulos estructurales del sistema y las relaciones entre los mismos.
- **Implementación:**
- Implementación del sistema experto. Se describe el proceso de implementación del sistema, así como todos los problemas surgidos durante el mismo y las soluciones aplicadas.
 - Implementación de la interfaz. Se describe el proceso de implementación de la interfaz, así como todos los problemas surgidos durante el mismo y las soluciones aplicadas.
- **Pruebas y validación:**
- Verificación. Se comprueba el funcionamiento y comportamiento del sistema.
 - Validación del sistema por un experto. Se comprueba que el funcionamiento del sistema es correcto.
- **Documentación:**
- Documentación del proyecto. Se procede a registrar y explicar todo el proceso de desarrollo así como las decisiones tomadas a lo largo del mismo.

1.3. Recursos humanos

Para poder desarrollar este proyecto he contado con la ayuda de varios músicos y compositores profesionales que me han ayudado a crear la base de conocimiento del sistema. Además, también he contado con profesores del Conservatorio Profesional de Música Ángel Barrios de Granada para poder llevar a cabo su validación.

Los expertos consultados han sido:

- Alberto José Moreno Montes (Violista y compositor)
- Isabel Salado Ortega (Pianista)
- Clara Luz Fernández Vecino (Profesora de armonía)

El tener varios expertos a los que consultar y especializados en diferentes áreas musicales hace que podamos construir una base de conocimiento mucho más amplia, consistente y mejor estructurada. Sin embargo, esto también supone una dificultad añadida. Ésta radica en el hecho de que, como se menciona en la introducción, algunos aspectos referidos a la música poseen un carácter subjetivo y cada experto puede tener opiniones o percepciones distintas sobre algunos conceptos o situaciones específicas. Teniendo en cuenta esto, tener una mayor cantidad de información y conocimientos que aprehender y seleccionar de cara a nuestro sistema puede suponer un proceso largo y complejo, incluso una “desventaja”, pero hace que el sistema sea mucho más completo y preciso a la hora de deducir y obtener conclusiones.

1.4. Temporización

Para percibir de forma más visual la planificación estructural y temporal en las figuras 1.4.1, 1.4.2 y 1.4.3 se muestran una serie de diagramas.

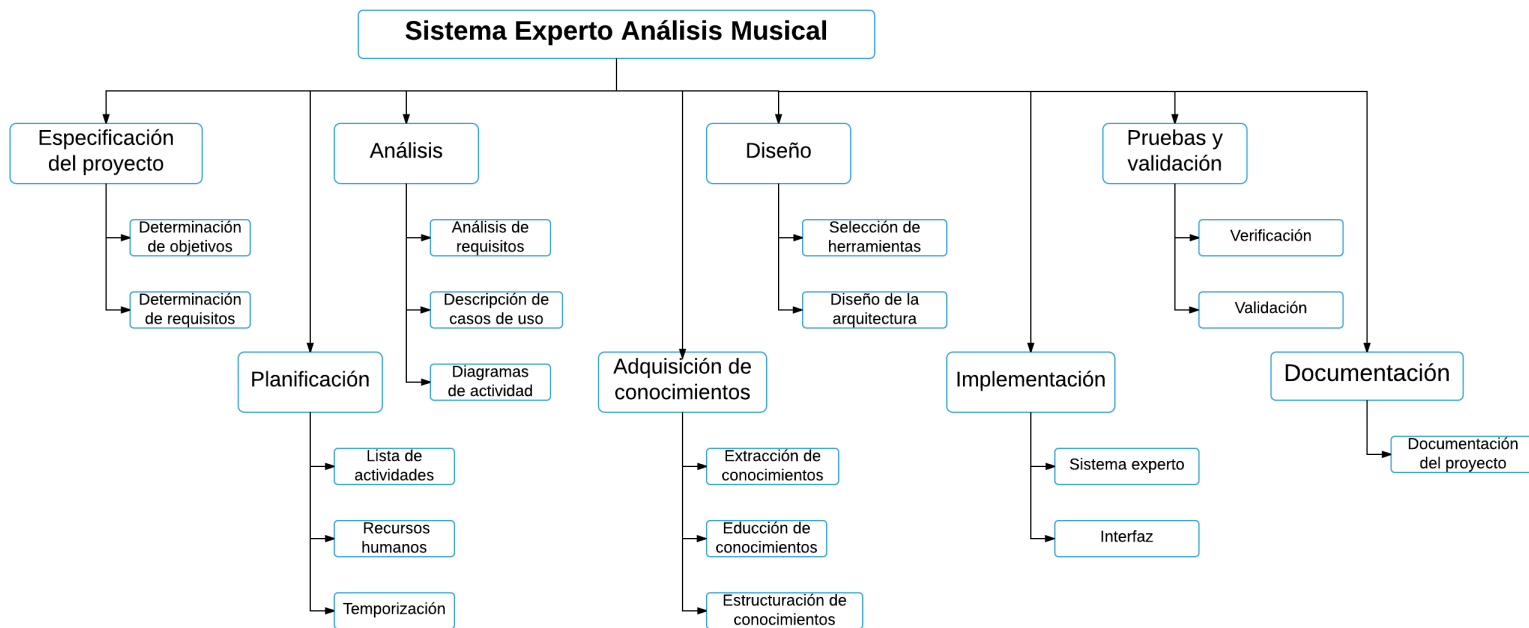


Figura 1.4.1: Diagrama de estructura de descomposición de trabajo

Nombre	Fecha de inicio	Fecha de fin
Especificaciones del proyecto	03/10/2016	06/10/2016
Planificación	07/10/2016	14/10/2017
Análisis de diseño	15/10/2016	31/10/2016
Adquisición del conocimiento	15/10/2016	22/05/2017
Implementación	21/01/2017	17/04/2017
Pruebas y validación	18/04/2017	07/05/2017
Documentación	08/05/2017	22/05/2017

Figura 1.4.2: Temporización de las tareas

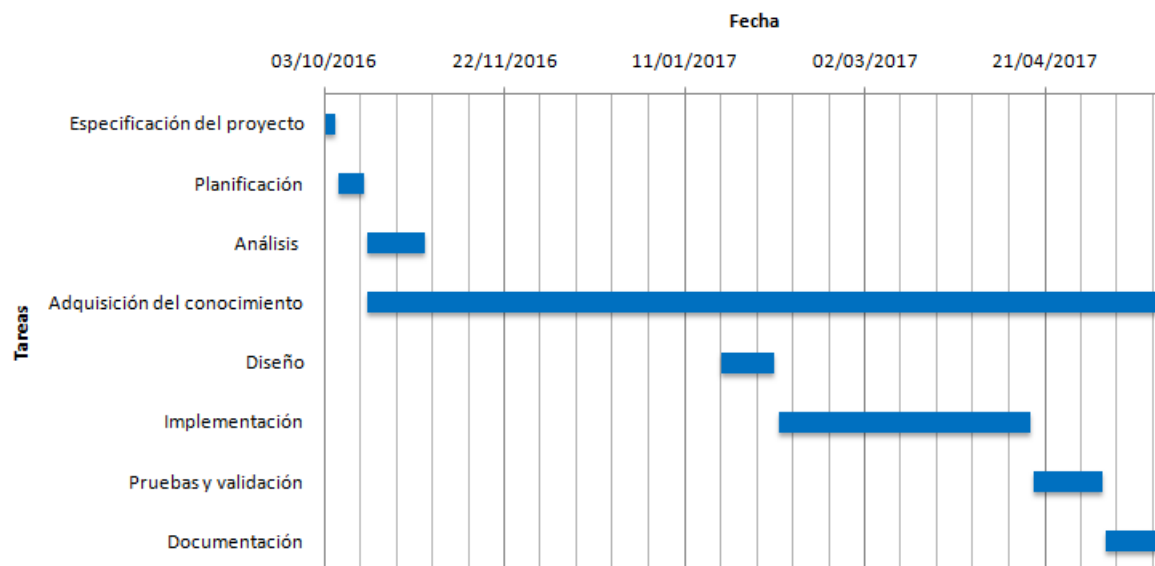


Figura 1.4.3: Diagrama de Gantt

2. Análisis

2.1. Objetivos

Los objetivos que se desean alcanzar con este sistema son los siguientes:

OBJ-1. El sistema deberá analizar una partitura en busca de faltas producidas por los movimientos armónicos de las voces.

OBJ-2. El sistema deberá analizar desde un punto de vista melódico una partitura y comprobar la existencia de errores referidos a la línea melódica y de contrapunto.

OBJ-3. El sistema deberá analizar desde un punto de vista armónico una partitura y comprobar la existencia de errores referidos a la armonía.

OBJ-4. El sistema mostrará todos los errores encontrados.

2.2. Análisis de requisitos

La especificación de requisitos de un sistema experto comprende la determinación de requerimientos de información, funcionales y de entrada de datos. En esta fase del desarrollo debemos determinar qué datos de entrada, proporcionados por el usuario, necesita el sistema para funcionar, las operaciones que llevará a cabo sobre dichos datos y los resultados que espera obtener del sistema.

Para poder realizar esta fase del desarrollo, necesitamos extraer esta serie de requisitos de los que serán los usuarios finales del sistema.

Como se menciona en la introducción, este sistema está dirigido a compositores o estudiantes de composición y armonía que desean mejorar sus obras o comprobar si éstas no contienen errores armónicos y/o melódicos. Éstos serían nuestros usuarios finales y a los que he consultado qué respuesta esperarían del sistema, es decir, qué elementos deberían analizarse en la partitura y cómo mostrar los posibles errores encontrados, así como qué información sería necesaria proporcionar para poder llevar a cabo el análisis deseado.

2.2.1. Actores

Los actores implicados serán tres: el **ingeniero del conocimiento** y desarrollador, los **expertos** y los **usuarios** finales.

- El **ingeniero del conocimiento** y desarrollador será el encargado de extraer y educir el conocimiento necesario para poder construir el sistema experto y llevar a cabo su implementación.
- Los **expertos** serán las personas que nos proporcionarán la principal fuente de conocimiento sobre el ámbito del problema a resolver por el sistema experto.
- Los **usuarios** finales serán las personas a las que estará dirigido el sistema para su uso. Además, también serán una fuente de conocimiento para el sistema experto, concretamente sobre lo relacionado a los requisitos y funcionalidades del sistema.

2.2.2. Requisitos de información

Requisitos de entrada de datos

Los datos de entrada necesarios para el funcionamiento del sistema son las opciones de los distintos tipos de análisis posibles y la partitura en la que se van a realizar.

Las opciones se mostrarían en un formulario multi-opción para permitir que el usuario pueda escoger una o varias según sus necesidades.

El formato de la partitura sería xml. Esto se debe a que al ser un estándar permite trabajar con la partitura de forma más cómoda y sencilla facilitando en gran medida el desarrollo del sistema. Además, todos los editores de partituras permiten exportar a este formato, con lo que no supondría tampoco un inconveniente para los usuarios.

- **RI-1. PARTITURA.**
 - Archivo que contiene la partitura de coro a analizar por el sistema.
 - **Requisitos asociados:** RF-1, RF-2.
- **RI-2. OPCIONES DE ANÁLISIS.**
 - Lista de opciones sobre los tipos de análisis posibles a realizar sobre la partitura en busca de errores.
 - **Requisitos asociados:** RF-1, RF-2.

Requisitos de salida de datos

El sistema, una vez finalizado el análisis, mostrará una lista seriada de los diferentes errores y faltas encontrados. Éstos estarán debidamente explicados y señalizados en la partitura, indicando las voces implicadas y el compás en el que aparecen. Dado que estos errores pueden ser de origen diverso - armónico o melódico - se crearán listados específicos para cada tipo de análisis, a fin de facilitar el entendimiento de los resultados y la experiencia del usuario.

■ RI-3. FALTAS POR MOVIMIENTOS.

- Lista de errores provocados por movimientos directos o paralelos de las voces.
- **Requisitos asociados:** RF-1.1, RF-1.2, RF-1.3, RF-1.4, RF-1.5, RF-1.6.

■ RI-4. ERRORES MELÓDICOS.

- Lista de errores referidos al contrapunto y la dirección de las voces.
- **Requisitos asociados:** RF-2.

■ RI-5. ERRORES ARMÓNICOS.

- Lista de errores referidos al mal uso de estructuras armónicas.
- **Requisitos asociados:** RF-1.7, RF-1.8, RF-1.9.

2.2.3. Requisitos funcionales

El eje principal de este sistema radica en dos funciones principales: análisis armónico y análisis melódico.

La realización del análisis armónico implica la comprobación de los siguientes hechos:

- *Quintas paralelas:* dadas dos voces, éstas no pueden producir dos intervalos de quinta consecutivos. No obstante, si el segundo intervalo de quinta es de tipo aumentado o disminuido y no se encuentra en voces extremas no se considerará falta.

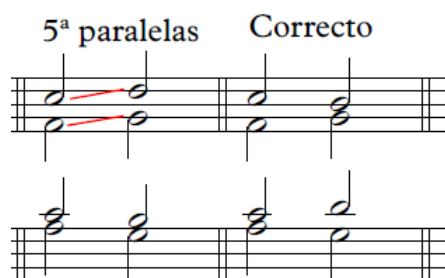


Figura 2.2.1: Ejemplo de quintas paralelas

- *Quintas directas*: dadas dos voces, éstas no pueden producir un intervalo de quinta por movimiento directo.

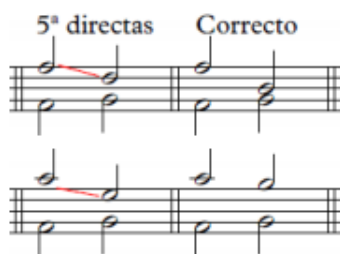


Figura 2.2.2: Ejemplo de quintas directas

- *Octavas paralelas*: dadas dos voces, éstas no pueden producir dos intervalos de octava consecutivos.

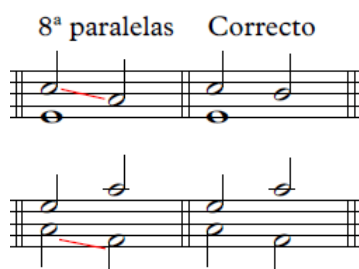


Figura 2.2.3: Ejemplo de octavas paralelas

- *Octavas directas*: dadas dos voces, éstas no pueden producir un intervalo de octava por movimiento directo.

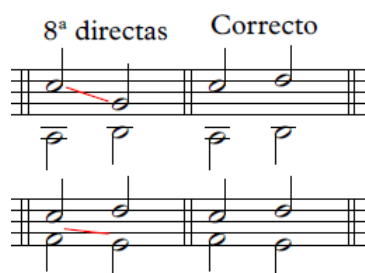


Figura 2.2.4: Ejemplo de octavas directas

- *Tritono*: dadas dos voces, éstas no pueden producir un intervalo de cuarta aumentada o tritono. No obstante, si éste viene preparado por movimiento conjunto u oblicuo de las voces no se considerará falta.



Figura 2.2.5: Ejemplo de cuarta aumentada

- *Duplicación de sensible*: en ningún caso se podrá duplicar la sensible de la tonalidad.



Figura 2.2.6: Ejemplo de duplicación de sensible

- *Búsqueda de acordes incompletos*: todos los acordes deberán tener la fundamental, la tercera y la quinta del acorde en al menos una voz, especialmente

si están invertidos. No obstante, se puede dar la situación de que el acorde de tónica quede incompleto, con la ausencia de la quinta, en la cadencia.

- *Segunda inversión consecutiva*: en ningún caso podrá haber dos acordes en segunda inversión consecutivos.
- *Lógica tonal*: la armonía de la obra deberá seguir, en la medida de lo posible, el esquema indicado en la figura .
 - La tónica o primer grado (I) podrá moverse hacia cualquier otro grado de la escala.
 - La supertónica o segundo grado (II) podrá moverse hacia el cuarto, quinto o séptimo grado.
 - La mediantes, modal o tercer grado (III) podrá moverse hacia sexto grado.
 - La subdominante o cuarto grado (IV) podrá moverse hacia el segundo, quinto o séptimo grado.
 - La dominante o quinto grado (V) podrá moverse hacia el primer, sexto o séptimo grado.
 - La superdominante o sexto grado (VI) podrá moverse hacia el segundo o cuarto grado.
 - La subdominante o séptimo grado (VII) podrá moverse hacia el primer, tercer o quinto grado.

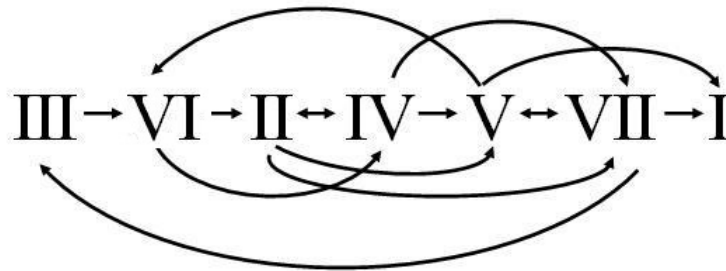


Figura 2.2.7: Diagrama de lógica tonal

Por otra parte, no podrán darse en ningún caso las siguientes sucesiones de acordes:

- I-II en estado fundamental.
- II-III en estado fundamental,

- III-IV en estado fundamental.

La realización del análisis melódico implica la comprobación de los siguientes hechos:

- *Resolución de sensibles*: la sensible de la tonalidad deberá resolver siempre en la tónica.

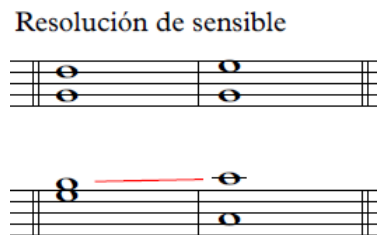


Figura 2.2.8: Ejemplo sensible sin resolver

- *Resolución de séptimas*: la séptima de dominante deberá resolver siempre por movimiento por grado conjunto descendente y deberá estar preparada.

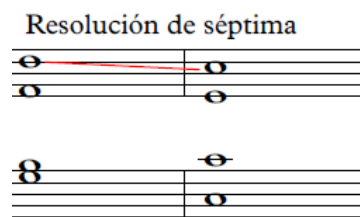


Figura 2.2.9: Ejemplo de séptima de dominante sin resolver

- *Segunda aumentada*: en ningún caso podrá haber un intervalo de segunda aumentada entre dos notas consecutivas en una misma voz.

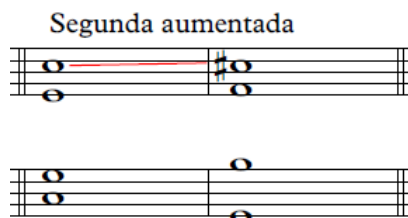


Figura 2.2.10: Ejemplo de segunda aumentada melódica

- *Tritono melódico*: en ningún caso podrá haber un intervalo de cuarta aumentada o tritono entre dos notas consecutivas en una misma voz.



Figura 2.2.11: Ejemplo de cuarta aumentada melódica

- *Contrapunto en voces extremas*: deberá predominar el movimiento contrario en las voces extremas sobre el movimiento oblicuo y, especialmente, directo.
- *Contrapunto del salto*: si hay dos movimientos (saltos) en la misma dirección, ascendente o descendente, la primera y la última nota no pueden ser disonantes.
- *Melodía incoherente*: deberá predominar el movimiento por grados conjuntos en las melodías, evitando hacer demasiados saltos.

En base a todas estas reglas los requisitos funcionales sería los siguientes:

RF-1.Análisis armónico.

El sistema llevará cabo un análisis sobre la armonía y los acordes de la partitura en busca de faltas y errores en las secuencias armónicas.

- **RF-1.1.** Se realizará la búsqueda de quintas paralelas.
- **RF-1.2.** Se realizará la búsqueda de quintas directas.
- **RF-1.3.** Se realizará la búsqueda de octavas paralelas.
- **RF-1.4.** Se realizará la búsqueda de octavas directas.
- **RF-1.5.** Se realizará la búsqueda de tritonos o cuartas aumentadas.
- **RF-1.6.** Se realizará la búsqueda de sensibles duplicadas.

- **RF-1.7.** Se realizará la búsqueda de acordes incompletos.
- **RF-1.8.** Se realizará la búsqueda de secuencias de acordes en segunda inversión consecutivos.
- **RF-1.9.** Se realizará una comprobación sobre las secuencias de acordes para asegurar que siguen las reglas de la lógica tonal.

RF-2. Análisis melódico.

El sistema llevará a cabo un análisis sobre las melodías de las voces en busca de una mala conducción de éstas o de un mal uso del contrapunto.

- **RF-2.1.** Se comprobará la resolución de sensibles en la tónica.
- **RF-2.2.** Se comprobará la resolución de todas las séptimas de dominante.
- **RF-2.3.** Se comprobará la existencia de intervalos de segunda aumentada melódicos.
- **RF-2.4.** Se comprobará la existencia de tritonos o intervalos de cuarta aumentada melódicos.
- **RF-2.5.** Se comprobará la existencia de un buen contrapunto entre las voces de la soprano y del bajo.
- **RF-2.6.** Se comprobará la existencia de saltos consecutivos con disonancias.
- **RF-2.7.** Se comprobará la existencia de una buena praxis en las líneas melódicas.

2.2.4. Requisitos no funcionales

Los requisitos no funcionales son aquellos referidos al desarrollo pero no necesariamente relacionados con la funcionalidad del sistema, como aspectos referidos a la interfaz, al rendimiento o a la fiabilidad del sistema.

RNF-1. Los usuarios del sistema deberán estar familiarizados con el lenguaje específico empleado en el ámbito musical para poder entender las explicaciones aportadas por el sistema.

RNF-2. Las pruebas y validaciones del sistema se llevarán a cabo por uno de los expertos.

2.3. Modelos de casos de uso

2.3.1. Descripción básica de los actores

Actor	Ingeniero del conocimiento	ACT-1
Descripción	Ingeniero encargado del proceso de adquisición de conocimiento y creación del sistema	
Características	Es también el desarrollador que lleva a cabo la implementación del sistema.	
Relaciones	ACT-2, ACT-3	
Referencias		

Tabla 2.3.1: Descripción del actor Ingeniero del Conocimiento

Actor	Experto	ACT-2
Descripción	Persona a la que el ingeniero del conocimiento entrevista para conocer el ámbito del problema y los procedimientos necesarios para su resolución	
Características	Persona experta en el ámbito del problema a resolver por el sistema experto.	
Relaciones	ACT-1	
Referencias		

Tabla 2.3.2: Descripción del actor Experto

Actor	Usuario	ACT-3
Descripción	Usuario final al que está dirigido el sistema	
Características	El usuario, al igual que el experto, tendrá algunos conocimientos sobre el dominio del problema	
Relaciones	ACT-1	
Referencias		

Tabla 2.3.3: Descripción del actor Usuario

2.3.2. Descripción de casos de uso

Caso de uso	Analizar partitura	CU-1
Actores	Usuario	
Tipo	Primario, esencial.	
Referencias	RI-1,RI-2,,RI-3,RI-4,RI-5,RF-1,RF-2	
Precondición	Partitura almacenada en el servidor y opciones seleccionadas enviadas	
Postcondición	Se crearán archivos con los resultados del análisis para mostrarse a través de la interfaz.	
Propósito	Realización del análisis deseado en la partitura aportada por el usuario y obtención de resultados.	
Resumen	Una vez se han proporcionado todos los datos necesarios, el usuario podrá activar el sistema para obtener la lista de errores y faltas encontrados en su partitura para su posterior corrección.	

Tabla 2.3.4: Descripción del caso de uso CU-1

Curso normal	
Accion del actor	Acción del sistema
(1) Usuario: selecciona las opciones de análisis propuestas por el sistema.	
	(2) Sistema: comprueba las opciones seleccionadas.
(3) Usuario: selecciona el archivo de la partitura y lo sube al servidor del sistema.	
	(4) Sistema: comprueba el archivo adjuntado.
(5) Usuario: envía los datos al sistema.	
	(6) Sistema: almacena las opciones seleccionadas por el usuario.
	(7) Sistema: almacena la partitura en el servidor.
	(8) Sistema: lanza los módulos de análisis seleccionados.
	(9) Sistema: obtiene los resultados de los módulos.
	(10) Sistema: muestra los resultados de los módulos.

Tabla 2.3.5: Descripción del curso normal del caso de uso CU-1

Cursos alternos
(2a) El usuario no ha seleccionado ninguna opción. Sistema: informa del error y vuelve a solicitar la selección de opciones.
(4a) El usuario no ha seleccionado ningún archivo. Sistema: informa del error y vuelve a solicitar la selección de un archivo.
(4b) El usuario ha seleccionado un archivo no válido. Sistema: informa del error y vuelve a solicitar la selección de un archivo válido.

Tabla 2.3.6: Descripción de los cursos alternos del caso de uso CU-1

2.4. Diagramas de actividad

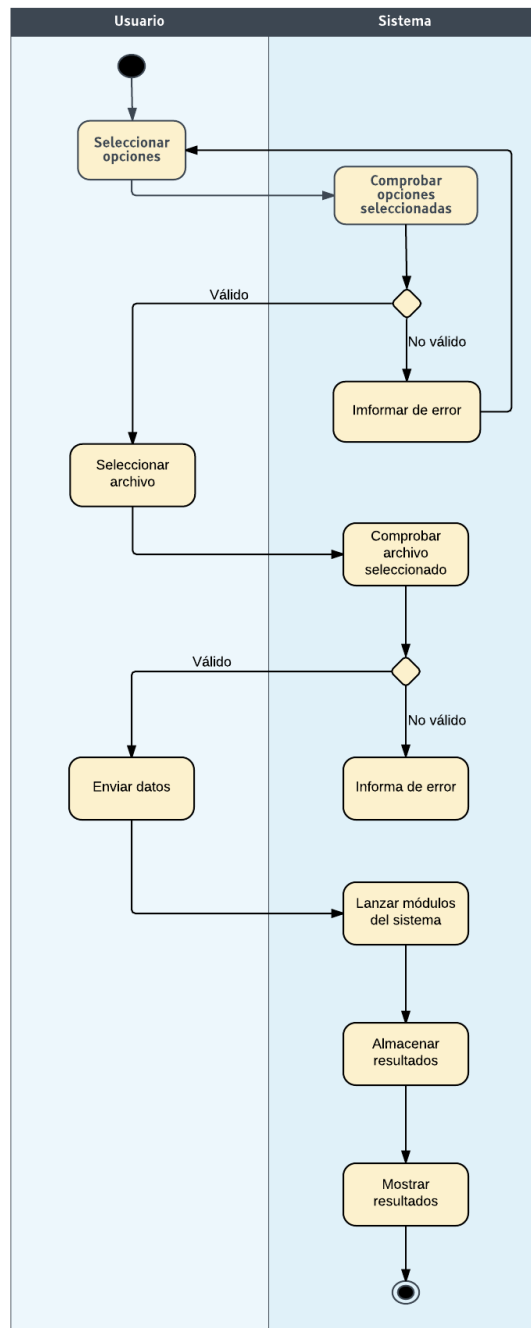


Figura 2.4.1: Diagrama de actividad del caso de uso CU-1. Analizar partitura.

3. Adquisición de conocimiento

La adquisición y educación del conocimiento constituye una de las fases más importantes del desarrollo de este tipo de sistemas. En esta etapa es donde el Ingeniero del Conocimiento interactúa con el experto para obtener la información sobre la solución de los problemas, así como las estrategias utilizadas para la obtención de cada solución. Además de consultar con el experto, también se adquiere conocimiento de otras fuentes como libros, tratados, estudios, etc. Este proceso supone el “cuello de botella” del desarrollo.

Esta tarea no es un paso concreto en la metodología de desarrollo, sino que se va produciendo en paralelo al resto de etapas de construcción del sistema.

El proceso de adquisición de conocimiento se ha llevado a cabo en tres fases:

- Extracción de conocimientos
- Educación de conocimientos
- Estructuración de conocimientos

Para llevar a cabo estas tareas, he consultado a los expertos Alberto José Moreno Montes e Isabel Salado Ortega, ambos músicos titulados.

3.1. Extracción de conocimientos

En la realización de esta tarea, se pretende aprender sobre el dominio tanto como sea posible antes de comenzar las sesiones con los expertos para facilitar la educación de conocimientos.

Al haber estudiado música en el conservatorio hace algunos años, esta fase supuso más bien un proceso de recordar que de aprender. No obstante, al ser un dominio tan extenso, es necesario consultar la documentación adecuada y más útil para la aplicación a nuestro problema. Por este motivo, pregunté a ambos expertos, Alberto e Isabel, sobre el material más importante a consultar, incluyendo libros, manuales y tratados. Las dos fuentes más relevantes y de las cuáles se extrajo la mayoría de conocimientos previos fueron el *Tratado de Armonía*, volúmenes I y II, de Joaquín Zamacois [Zamacois, 2002a, Zamacois, 2002b] y *Armonía* por Walter Piston [Piston, 2007]. Ambos se incluyen, junto con el resto de fuentes, en las referencias bibliográficas.

De estos libros se determinaron la mayoría de definiciones y conceptos generales sobre lenguaje musical y armonía.

3.2. Educción de conocimientos

Gracias a los conocimientos adquiridos en la fase anterior, realizamos una serie de entrevistas estructuradas con los expertos. En estas entrevistas se formularon cuestiones concretas y previamente planificadas con el fin de educir conocimiento de una forma más eficiente. Nos centramos en definir conceptos específicos y procedimientos necesarios a la hora de analizar partituras, haciendo especial énfasis en el análisis melódico, el cual tiende a ser más delicado de realizar y con un componente subjetivo más fuerte.

La mayor dificultad de este proceso radicó en la cantidad de situaciones excepcionales que pueden darse y de la subjetividad que viene adherida a cualquier arte. Por ejemplo, las excepciones a la falta de *quintas paralelas*. Las *quintas paralelas* son una de las faltas más graves, dado que al darse se produce una sonoridad más dura y arcaica. Sin embargo, éstas pueden permitirse si la segunda quinta que se forma es una quinta disminuida, salvo que ésta se forme con el bajo o en voces extremas, en cuyo caso se considerará falta.

Como podemos observar en este ejemplo, las reglas armónicas no son directrices sencillas de llevar a cabo o entender fácilmente. Con el fin de poder comprenderlas empleé con Alberto e Isabel lo que denominamos “incidentes críticos”.

3.2.1. Incidentes críticos

Con la técnica de “incidentes críticos”, ambos expertos me describieron los casos excepcionales mencionados anteriormente y algunas de las dificultades más relevantes a la hora de realizar un análisis armónico y/o melódico.

Me explicaron todas las posibles excepciones para cada regla, ejemplificándome cada una de las situaciones.

En el caso de los problemas derivados por la subjetividad a la hora de analizar las melodías de un coral, Alberto sugirió ceñirse lo más posible al estilo de los corales compuestos durante la época barroca, ya que las restricciones a la hora de componer eran más fuertes y estaban mejor delimitadas.

3.3. Estructuración de conocimientos

Una vez hemos finalizado la extracción y la educación de conocimientos pasamos a analizar, organizar y estructurar todos los conocimientos obtenidos para crear nuestra base de conocimiento.

En un primer paso, contrastamos toda la información que hemos adquirido junto con los requisitos del sistema para filtrar qué conocimientos son los necesarios y útiles para poder llevar a cabo los distintos análisis en el coral. Los tipos de análisis son independientes entre sí, por lo que se hizo una filtración de información para cada uno de ellos. Aunque el tener distintos análisis sobre la partitura podía suponer una dificultad para este proceso de selección, el hecho de ser independientes supuso una gran ventaja, facilitándolo en gran medida.

A continuación, organizamos y estructuramos el conocimiento dividiéndolo en *hechos* y *reglas*.

- Los *hechos* serán asertados en el sistema. Para definir la mayoría de estos hechos se hará uso de estructuras “**templates**” o plantillas.

```
(intervalo (distancia d) (tipo t) (nota1 n1) (nota2 n2))
```

- Las *reglas* definirán los requisitos funcionales del sistema en forma de estructuras *si «antecedente» entonces «consecuente»*.

```
if intervalo.distancia = 2 then intervalo.tipo = disonante
```

Este conjunto de hechos y reglas constituirá la base de conocimiento del sistema experto. Estas reglas utilizarán los hechos almacenados en la base de conocimiento, junto con los hechos calculados a partir de los datos de entrada, para deducir nuevos hechos y conocimientos que permitan obtener conclusiones finales y así obtener resultados.

3.3.1. Hechos

Los hechos del sistema se dividen en tres tipos:

- **Hechos asertados:** son los hechos que constituyen la base de conocimiento “inicial” del sistema. Estos hechos constituyen todo el conocimiento necesario ,referido principalmente a conceptos, para poder analizar una partitura.
- **Hechos calculados:** son los hechos que extraemos directamente de la partitura, es decir, las notas que la componen con sus alteraciones, su altura y su duración.
- **Hechos deducidos:** son los hechos que el sistema deduce aplicando las reglas a los hechos asertados y calculados. También puede deducir hechos a partir de hechos previamente deducidos que sea utilizados por las reglas.

Los hechos asertados están referidos a los siguientes conceptos:

- *Tonalidades:* todas las posibles tonalidades existentes indicando el nombre de la tonalidad, el modo, las alteraciones que tienen en su armadura y la sensible.

(tonalidad (nombre D) (tipo mayor) (armadura 2s) (sensible Cs))

Las tonalidades las usaremos para poder determinar qué tipo de alteraciones tienen las notas de la partitura por defecto y cuáles serían alteraciones accidentales. También nos indican la sensible de la tonalidad, la cuál usaremos para comprobar varios errores.

- *Escalas:* todas las escalas mayores y menores asociadas a cada tonalidad, indicando el nombre de la escala y las notas (grados) que la componen.

(escala (nombre D) (tipo mayor) (grado_I G) (grado_II A) (grado_III B)
(grado_IV C) (grado_V D) (grado_VI E) (grado_VII Fs))

A partir de la escala podemos deducir todos los acordes fundamentales de la tonalidad. Con ellos realizaremos el análisis armónico comprobando acordes incompletos, sucesiones erróneas, etc.

- *Intervalos:* todos los tipos de intervalos desde segundas menores hasta octavas justas desde cada una de las posibles notas. Indicamos la distancia, el tipo y las dos notas que lo forman.

(intervalo (distancia 3) (tipo mayor) (nota1 G) (nota2 B))

Necesitamos conocer todos los intervalos posibles para poder evaluar saltos y contrapunto en las melodías, así como movimientos armónicos entre las voces, como las quintas paralelas u octavas entre otras faltas.

- *Acordes*: todos los acordes de cada uno de los grados de cada tonalidad. Los acordes pueden ser de tríada (formados por tres notas) o cuatríada (formados por cuatro notas), por lo que diferenciamos ambos tipos. Indicamos el grado de la escala a partir de la cuál se forma el acorde, el nombre y el modo de la tonalidad a la que pertenece y las notas que lo componen: fundamental, tercera y quinta en el caso de acordes de tríada y fundamental, tercera, quinta y séptima en el caso de acordes de cuatríada.

```
(acorde_3 (grado I) (nombre_tonalidad C) (modo_tonalidad mayor) (
    fundamental C) (tercera E) (quinta G))
```

```
(acorde_4 (grado V) (nombre_tonalidad C) (modo_tonalidad mayor) (
    fundamental G) (tercera B) (quinta D) (septima F))
```

Los acordes los utilizaremos para poder realizar el análisis armónico.

- *Sucesiones lógicas*: lista de sucesiones permitidas entre acordes según las normas de la armonía tradiciones. Se indican los dos grados que componen una sucesión lógica.

```
(sucesion_logica (grado1 V) (grado2 I))
```

Esto nos permite analizar la coherencia armónica de la obra, es decir, que la estructura armónica sobre la que se construye el coral sea correcta y tenga una buena sonoridad.

Los hechos calculados del sistema son aquellos que extraemos a partir de la información contenida en la partitura.

- *Tonalidad*: se extrae la tonalidad de la obra para poder determinar sobre qué escala tenemos que trabajar y qué acordes son los que vamos a analizar.
- *Compás*: se extrae el compás para poder localizar los errores.
- *Notas*: se extrae la información sobre las notas de la partitura. Dependiendo de si se va a realizar un análisis melódico o armónico extraeremos dos listas distintas de notas.

- **Movimientos:** se extrae también información sobre los movimientos de las notas para poder analizar tanto errores armónicos como melódicos.

Toda la información referida a estos hechos se explicará con más detalle en el apartado de la implementación.

Por último, los hechos deducidos serán los que irá asertando el sistema tras aplicar reglas y utilizará para ir deduciendo nuevos hechos hasta llegar a las conclusiones finales, que en este caso serán la existencia o no de errores y faltas.

3.3.2. Reglas

Las reglas del sistema coinciden prácticamente con los requisitos funcionales descritos en el capítulo anterior, aunque para poder implementar algunas de estas reglas ha sido necesario añadir reglas auxiliares.

- *Reglas para cargar los hechos iniciales:* debido a varios problemas con la implementación, que se comentarán en el capítulo 5, los hechos asertados comentados en el apartado anterior serán asertados a partir de unas reglas que los cargarán en la memoria de trabajo.
- *Reglas para la lectura de los datos extraídos de la partitura:* a continuación cargamos los hechos calculados. Estos son la información referida a las notas de la partitura, que ha sido previamente procesada para facilitar la aserción de estos hechos.
- *Reglas para detección de quintas:* este conjunto de reglas se encarga de detectar errores de quintas paralelas y quintas directas, teniendo en cuenta posibles casos excepcionales.
- *Reglas para detección de octavas:* detectan errores de octavas paralelas y octavas directas, comprobando posibles casos excepcionales.
- *Reglas para detección de tritonos armónicos:* comprueban si se da alguna el intervalo de cuarta aumentada entre dos voces, teniendo en cuenta la excepción de si está o no preparado.

- *Regla para comprobar la duplicación de sensible:* comprueba que la sensible no se encuentre duplicada en dos voces bajo ninguna circunstancia.
- *Regla para comprobar la resolución de sensible:* esta regla comprueba que la sensible resuelva correctamente en la tónica de la tonalidad.
- *Regla para comprobar la resolución de séptima:* al igual que en el caso de la sensible, se comprueba que la séptima también resuelva correctamente, sin tener en cuenta ninguna resolución excepcional.
- *Reglas para comprobar tritonos y segundas aumentadas melódicas:* estas reglas comprueban que no se produzcan este tipo de intervalos en la melodía de ninguna voz.
- *Reglas para comprobar el contrapunto en voces extremas:* este conjunto de reglas analiza los movimientos de las melodías de la soprano y el bajo para comprobar que predomina el movimiento contrario u oblicuo sobre el movimiento directo entre las voces.
- *Reglas para comprobar el contrapunto del salto:* este conjunto de reglas realiza una función similar al contrapunto de voces extremas, pero se centra en cada línea melódica. Se comprueba que no se den saltos que puedan provocar disonancias.
- *Reglas para comprobar las líneas melódicas:* estas reglas analizan los movimientos de las voces respecto a si se hacen demasiados saltos grandes o predomina el movimiento por grado conjunto en las melodías. Con esto podemos determinar si una melodía está bien construida o por el contrario no tiene coherencia melódica.
- *Reglas para analizar acordes:* se analizan armónicamente el coral determinando su estructura armónica. Se consideran tanto acordes de tríada como acordes con séptima teniendo en cuenta sus posibles inversiones.
- *Reglas para analizar la lógica tonal:* a partir del análisis previo de los acordes, se comprueba que las sucesiones de éstos sean correctas y sigan patrones armónicos tradicionales.

- *Reglas para comprobar los acordes:* este conjunto de reglas comprueba que todos los acordes estén completos, es decir, que no falte ninguna de las notas que lo componen por hacer una mala duplicación de las notas en las voces.
- *Reglas para mostrar los errores:* una vez detectados todos los errores con las reglas anteriores, implementamos una serie de reglas para poder mostrar los errores al usuario. Cada error se muestra debidamente explicado y localizado en la partitura.

3.3.3. Estructura

En total el sistema está compuesto por 873 hechos iniciales, previamente asertados, y 174 reglas. Hay que tener en cuenta que estos hechos iniciales son los hechos asertados, sin tener en cuenta todos los hechos que serán extraídos de la partitura. Dependiendo de la longitud del coral, se asertarán más o menos hechos calculados. Podemos estimar la cantidad de hechos calculados referidos a las notas de la siguiente manera:

- Si el compás es binario:

$$\text{hechos_calculados_notas} = n^\circ \text{ de partes del compas} \times 16 \times n^\circ \text{ de compases}$$

- Si el compás es ternario:

$$\text{hechos_calculados_notas} = n^\circ \text{ de partes del compas} \times 8 \times n^\circ \text{ de compases}$$

Para hacernos una idea más exacta, si nuestro coral tiene 24 compases y un compás $\frac{4}{4}$ (compás binario de cuatro partes) en total tendríamos de 1536 hechos referidos únicamente a las notas. Tenemos que tener en cuenta que estos serían utilizados únicamente para realizar el análisis armónico. Para realizar el análisis melódico necesitamos el número de notas de cada voz, que dependerá del coral y no podemos conocer su número a priori. Además, tenemos otra serie de hechos calculados como los movimientos de las notas, que serán iguales a la cantidad de notas para el análisis melódico.

Esto significa que para analizar de manera completa un coral de la misma longitud y compás que el del ejemplo anterior y suponiendo que sea un coral sencillo sin demasiadas notas, vamos a suponer tres notas por compás en cada voz, contaríamos con cerca de 3000 hechos iniciales.

Para facilitar la implementación y hacerlo más usable, como se ha comentado anteriormente, el sistema estará formado por varios módulos. A continuación se muestra un listado de los módulos indicando los hechos y reglas que los componen y el número de estos, sin tener en cuenta el número de hechos calculados que dependerán de la partitura:

- **Módulo 1:** 408 hechos y 42 reglas.
 - Hechos asertados:
 - Tonalidades
 - Intervalos
 - Hechos calculados:
 - Tonalidad de la obra
 - Compás
 - Lista de notas de las voces para análisis armónico
 - Lista de los movimientos de las melodías
 - Reglas:
 - Reglas para cargar hechos iniciales
 - Reglas para cargar hechos calculados
 - Reglas para errores de quintas
 - Reglas para errores de octavas
 - Reglas para tritonos armónicos
 - Regla para duplicación de sensible
 - Reglas para mostrar errores
- **Módulo 2:** 858 hechos y 82 reglas.
 - Hechos asertados:
 - Tonalidades
 - Intervalos
 - Escalas
 - Acordes

- Hechos calculados:
 - Tonalidad de la obra
 - Compás
 - Lista de notas de las voces
 - Lista de los movimientos de las melodías
 - Reglas:
 - Reglas para cargar hechos iniciales
 - Reglas para cargar hechos calculados
 - Regla para resolución de sensible
 - Regla para resolución de séptima
 - Reglas para intervalos disonantes melódicos
 - Reglas para contrapunto en voces extremas
 - Reglas para contrapunto del salto
 - Reglas para comprobar líneas melódicas
 - Reglas para mostrar errores
- **Módulo 3:** 875 hechos y 50 reglas.
- Hechos asertados:
 - Tonalidades
 - Intervalos
 - Escalas
 - Acordes
 - Lógica tonal
 - Hechos calculados:
 - Tonalidad de la obra
 - Compás
 - Lista de notas de las voces para análisis armónico
 - Reglas:

- Reglas para cargar hechos iniciales
- Reglas para cargar hechos calculados
- Regla para analizar acordes
- Regla para comprobar la lógica tonal
- Reglas para buscar acordes incompletos
- Reglas para mostrar errores

Como se puede observar, el sistema consta de una base inicial de conocimiento bastante compleja y amplia, aún sin tener en cuenta el conocimiento que nos aporta la partitura, el cuál aumenta aún más la complejidad del problema. También debemos tener en cuenta la cantidad de hechos deducidos a partir de las reglas, pero está dependerá de la cantidad de errores y faltas que haya en la partitura.

En la siguiente figura se muestra un diagrama de la estructura del sistema experto para facilitar su comprensión:

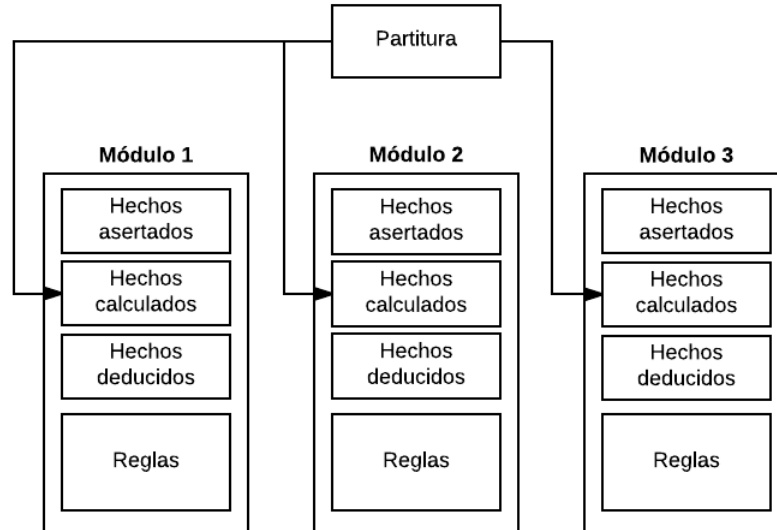


Figura 3.3.1: Diagrama de la estructura del sistema experto

4. Diseño

4.1. Selección de herramientas

4.1.1. Sistema experto

Para la implementación del sistema experto se consideraron las siguientes opciones:

- **SWI-PROLOG**. Lenguaje basado en lógica y programación declarativa caracterizado por su capacidad de manipulación simbólica.
- **Expert System Builder**. Programa gratuito, implementado en Common Lisp, destinado a simplificar el desarrollo de sistemas expertos.
- **CLIPS**. Lenguaje basado en reglas para desarrollar sistemas expertos. Basado en el lenguaje C para permitir una mayor portabilidad.

La herramienta Expert System Builder se desestimó al estar orientada al desarrollo de sistemas expertos para la toma de decisiones de una organización.

La decisión entre escoger CLIPS o SWI-PROLOG se determinó por los siguientes factores:

- CLIPS utiliza encadenamiento hacia delante y SWI-PROLOG encadenamiento hacia atrás. Los lenguajes que emplean encadenamiento hacia delante son más rápidos en el proceso de deducción, lo cuál hace que CLIPS sea más rápido en la ejecución de reglas que SWI-PROLOG.
- CLIPS se basa en el lenguaje C, el cuál sabemos es uno de los más eficientes en términos de velocidad computacional.
- Existen varias librerías para poder embeber sistemas expertos escritos en CLIPS en otros lenguajes como Java, Python o PHP.

Por lo tanto, el sistema se implementará utilizando el lenguaje CLIPS.

4.1.2. Interfaz

La primera decisión que había que tomar sobre la interfaz era si diseñar una aplicación web o una aplicación de escritorio para comunicarse con el sistema experto.

La aplicación de escritorio presenta las ventajas de ser más estable, robusta y eficiente en términos de velocidad.

La aplicación web presenta las ventajas de ser más accesible para los usuarios y una mayor portabilidad. Además, en caso de actualizar o mejorar el sistema, no habría ningún problema de incompatibilidad entre versiones.

Debido a estas características, se optó por diseñar una aplicación web, principalmente por las ventajas que ofrece de portabilidad y actualización, dado que una de las ventajas de los sistemas expertos es su escalabilidad.

En un principio se optó por PHP, pero las librerías necesarias para poder implementar el sistema experto en PHP se encontraban desactualizadas, por lo que escogimos como herramienta utilizar un framework web de Python, *Django*. Se escogió Django de entre los posibles frameworks para web de Python por su gran comunidad y documentación más accesible y completa. Además, cuenta con una portente interfaz de administración y permite un desarrollo ágil y rápido.

En las siguientes imágenes se muestra una idea inicial del diseño de las páginas de la interfaz:

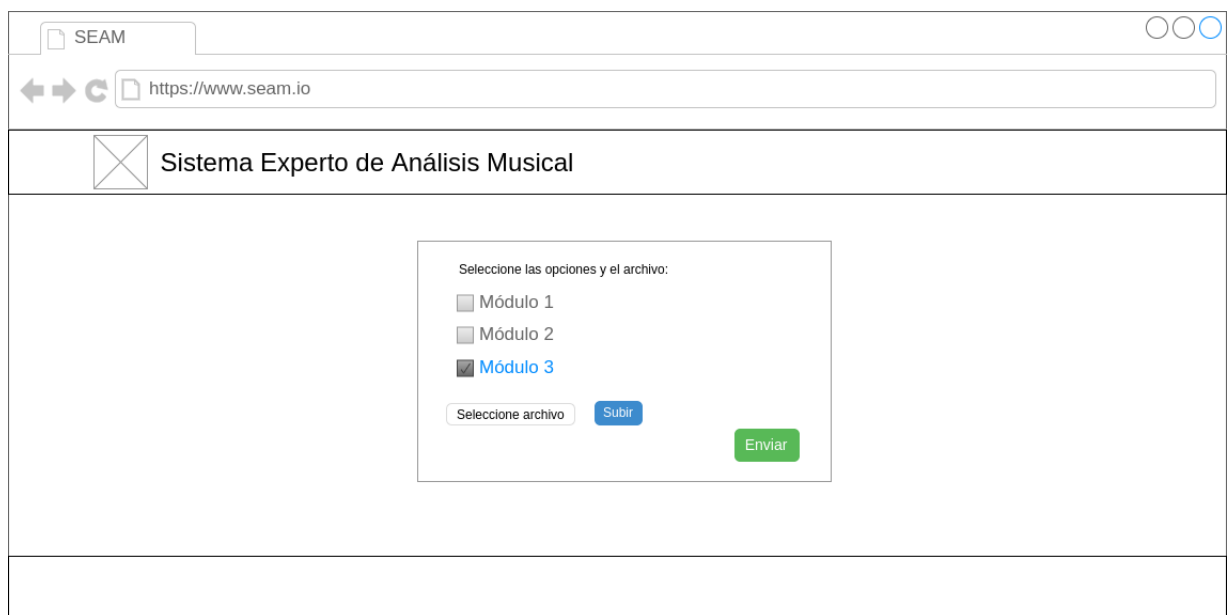


Figura 4.1.1: Wireframe de la página principal de la interfaz

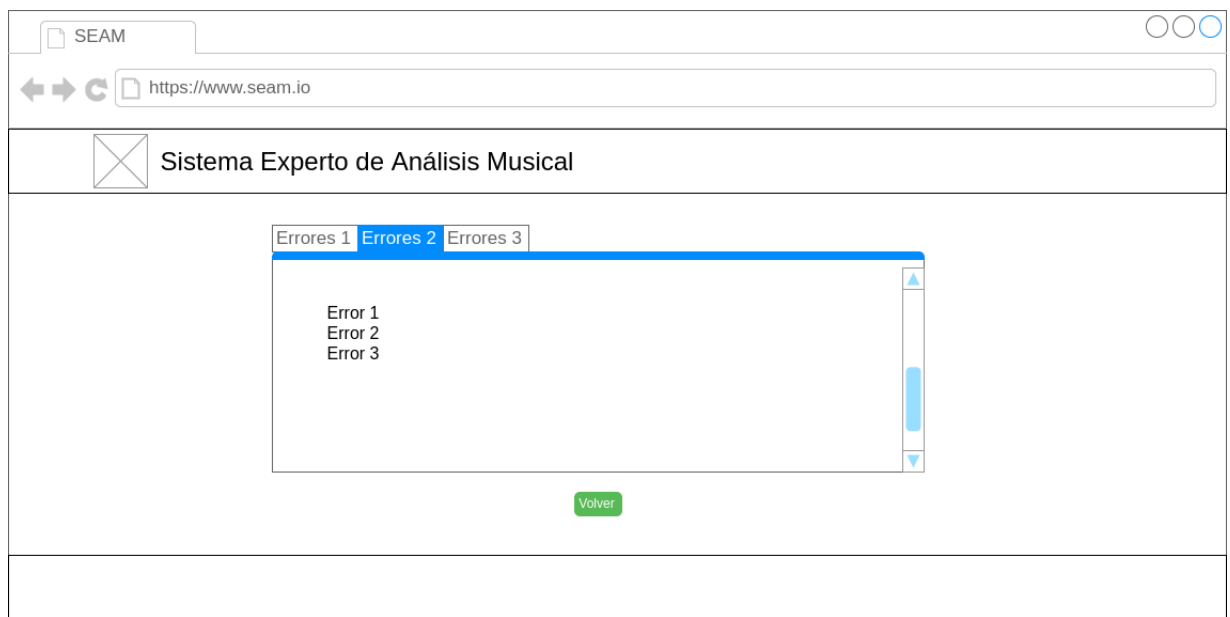


Figura 4.1.2: Wireframe de la página de resultados de la interfaz

La paleta de colores utilizada se muestra en la figura 4.1.3.



Figura 4.1.3: Paleta de colores de la interfaz

El uso de cada color se describe a continuación:

- El tono rojo se utilizará como fondo.
- Los tonos blanco y negro se utilizarán para el texto, y además el blanco para los fondos del formulario y la ventana de resultados.
- El tono gris claro se utilizará para resaltar algunos detalles.
- El tono gris oscuro se usará para la cabeza y el pie.

La maquetación de la interfaz se llevará a cabo haciendo uso de HTML5 y CSS3, apoyándonos en el sistema de templates Jinja2. Adicionalmente, se hará uso de jQuery para algunas funcionalidades.

4.2. Arquitectura del sistema

La arquitectura será **MVT** (Model-View-Template), que es la arquitectura empleada por Django. En esta arquitectura, las vistas hacen el papel de “controlador” del sistema. Éstas serán las que contendrán al sistema experto.

El sistema experto se dividirá en tres módulos como se comentó en el capítulo anterior:

- Módulo 1: realizará la búsqueda de faltas en los acordes provocadas por los movimientos de las voces, como las faltas de quintas paralelas, octavas paralelas o duplicación de sensibles.
- Módulo 2: realizará el análisis melódico para señalar todos los errores en las líneas melódicas y su contrapunto.
- Módulo 3: comprobará la lógica tonal del coral y comprobará las secuencias de acordes, así como que estos se encuentren completos.

En la imagen 4.2.1 se muestra gráficamente cómo sería la arquitectura del sistema.

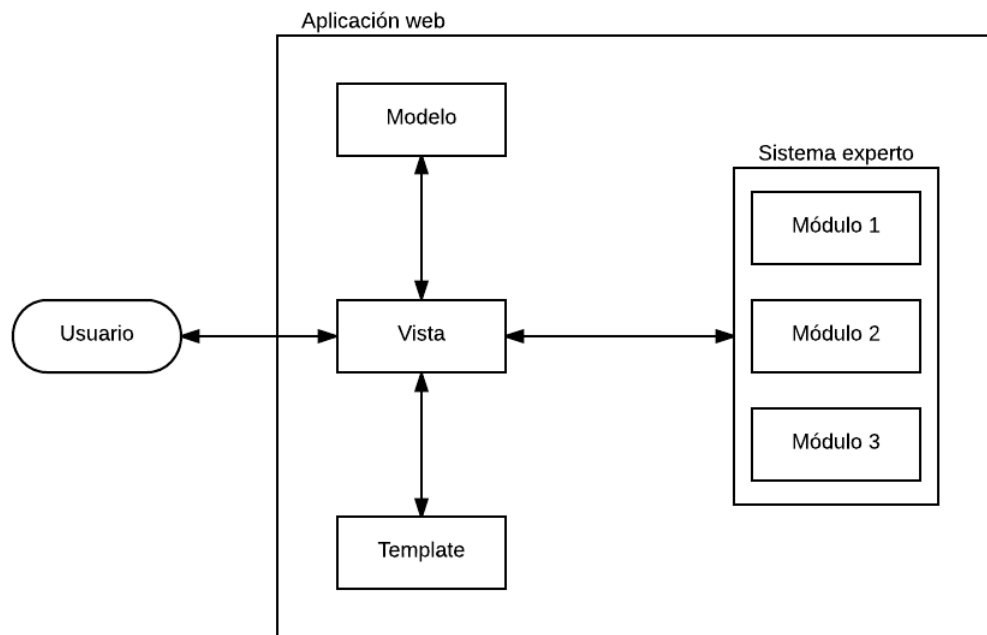


Figura 4.2.1: Diagrama de la arquitectura del sistema

5. Implementación

La implementación de este proyecto se ha llevado a cabo en dos fases diferenciadas: la creación del sistema experto y el desarrollo de la aplicación web, utilizada como interfaz, en la que será embebido.

Todo el código puede ser consultado en [López, 2017].

5.1. Sistema experto

El primer paso fue la creación de un prototipo utilizando el entorno de CLIPS. A continuación, se explican las decisiones tomadas a lo largo del desarrollo de este prototipo.

5.1.1. Formato de los datos de entrada: partitura

Una de las primeras cuestiones fue cómo obtener la información necesaria del XML en el que está contenida la partitura. Dado que CLIPS no puede procesar el XML para leer el contenido de las etiquetas, esta funcionalidad tenía que ser realizada por Python. Sin embargo, definimos las estructuras que debían tener los ficheros para poder ser leídos por CLIPS y añadir esa información en forma de hechos.

Los datos que teníamos que extraer de la partitura eran:

- La **tonalidad** de la obra. La tonalidad es la que nos define la *escala musical* sobre la que está construido el coral. Nos da la relación entre los acordes y sus funciones tonales. La tonalidad está compuesta por el nombre y el modo.
- El **compás**. El compás nos indica cuántas figuras de un determinado tipo hay en cada pulso. El compás está compuesto por dos números que nos indican el número de pulsos y el tipo de figura que dura un pulso:
 - 1: redonda
 - 2: blanca
 - 4: negra
 - 8: corchea

- Las **notas** de cada voz. Las notas musicales vienen definidas por su nombre, su altura, su duración y su alteración.
- Los **movimientos** de cada voz. Los movimientos nos indican si la voz, de una nota a otra, se mueve de forma ascendente, descendente o se mantiene.

Para cada uno de estos datos se creo una plantilla con la instrucción “**deftemplate**”, al igual que para el resto de hechos como se mencionó en el capítulo *Adquisición de conocimientos*.

Estructura del archivo de notas

Una de las mayores dificultades fue decidir la estructura del archivo que contendría la información sobre las notas. Cada nota tiene una duración independiente, lo cual a la hora de realizar el análisis armónico resulta muy complejo. Al analizar la armonía de una obra tenemos que tener en cuenta todas las voces al mismo tiempo, analizamos de forma vertical. Sin embargo, en una misma parte cada nota de cada voz puede tener una duración distinta y por ende puede formarse más de un acorde en dicha parte. Esto significa que no siempre va a haber el mismo número de acordes, ni de notas, en cada compás.

En la práctica, a un experto este hecho no le resulta un problema, porque puede observar las voces gráficamente y poder determinar cuándo termina la duración de cada nota y el acorde. En cambio, para un sistema experto, al no saber dónde termina cada nota con respecto a las demás, no puede hacer esas deducciones de manera tan sencilla.

Para solucionar este problema, decidí subdividir todas las notas de todas las voces a semicorcheas, que es la figura musical más pequeña que se suele utilizar en los corales. De esta forma, conseguimos que en todos los compases haya el mismo número de notas en cada voz.

En la siguiente figura, se muestra gráficamente lo que supone esta transformación:

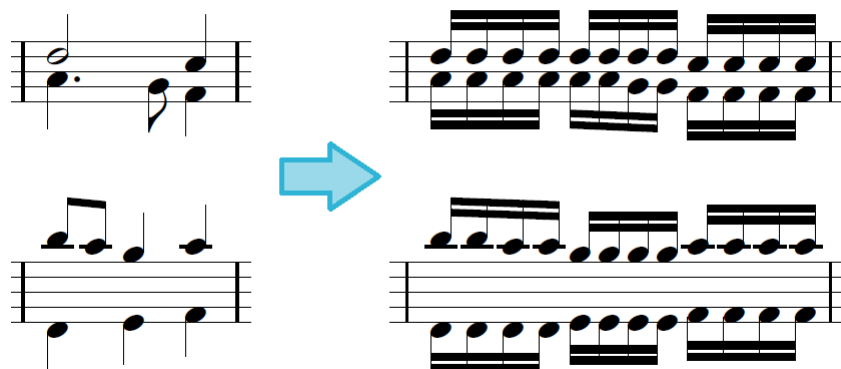


Figura 5.1.1: Subdivisión de un compás del coral “Aus meines Herzens Grunde” de *J.S.Bach* a semicorcheas

Como podemos ver, al realizar esta subdivisión podemos ir analizando en bloques equivalentes en tiempo, verticalmente es más sencillo analizar de esta manera que sin la subdivisión. Los resultados del análisis no se ven afectados, dado que únicamente hemos modificado la duración de las notas, la cual no interviene en los errores que vamos a detectar.

En el caso del análisis melódico, no es necesario hacer esta subdivisión porque analizamos cada voz de manera independiente de las otras, analizamos la partitura horizontalmente.

Formatos finales

Las estructuras de los ficheros que leería el sistema experto quedarían de la siguiente manera:

- Fichero de la tonalidad:

Nombre Modo

Ejemplo:

D Mayor

- Fichero del compás:

Partes Figura

Ejemplo:

3 4

- Fichero de notas: se crearía un fichero para cada voz.

Nombre Alteracion Altura Nombre Alteracion Altura ...

Ejemplo:

A b 4 B 2 D s 3

- Movimientos:

Movimiento1 Movimiento2 Movimiento3 ...

Ejemplo:

Ascendente Ascendente Descendente

Los nombres de las notas se representarán con el cifrado americano, el empleado en el archivo XML. La equivalencia entre el cifrado americano y el empleado en occidente es:

- Do \rightarrow C
- Re \rightarrow D
- Mi \rightarrow E
- Fa \rightarrow F
- Sol \rightarrow G
- La \rightarrow A
- Si \rightarrow B

5.1.2. Implementación del primer prototipo

En el primer prototipo, no se modularizaron los hechos y reglas en función de los módulos descritos en los capítulos anteriores. Esto es porque aún no estaban definidas del todo las dependencias de las reglas respecto a los hechos ni la división del análisis armónico en dos partes realmente diferenciadas.

Aunque en el análisis y el diseño quedan claras estas divisiones, a la hora de implementar las reglas, iban surgiendo muchas relaciones entre todos los hechos de la base de conocimiento. Por ejemplo, se esperaba que los hechos referidos a los movimientos de las voces únicamente fueran utilizados por el módulo del análisis melódico. Sin embargo, también es necesario para localizar algunos errores armónicos. Además, la existencia de algunos errores melódicos provoca a su vez errores armónicos y viceversa.

Una vez se terminaron de implementar todas las reglas y hechos, se procedió a verificar y validar el prototipo.

La **verificación** consistió en comprobar que los ficheros se leían correctamente y los hechos deducidos se asertaban en el sistema de la manera esperada. Esto se comprobó fácilmente observando la ventana de hechos que aparece en la propia interfaz de CLIPS.

La **validación** se realizó creando archivos de prueba, con los formatos descritos anteriormente, que contuviesen al menos un error de cada tipo. Los modelos de prueba fueron recogidos de las fuentes bibliográficas, las cuáles contienen una gran cantidad de ejemplos.

5.1.3. Modularización del sistema

Tras la creación del primer prototipo y su verificación y validación, comencé el proceso de modularización del sistema.

Al estar implementadas todas las reglas y funcionalidades del sistema, este proceso, aunque tedioso, fue el más sencillo.

Fui comprobando para cada regla los antecendentes que debían cumplirse y así poder agruparlas según los hechos que necesitaban. Una vez agrupadas, pude definir los hechos asertados necesarios para cada uno de los módulos.

Como resultado final, obtuve los tres módulos de análisis, cada uno con sus reglas y hechos diferenciados, tal y como se describen en el capítulo de Adquisición del conocimiento y que puede verse en 3.3.1.

5.2. Aplicación web

La implementación de la aplicación web, como se comentó en el capítulo anterior, se ha hecho utilizando el framework Django de Python.

La finalidad de implementar esta aplicación es proporcionar al sistema experto una interfaz agradable y sencilla de utilizar para el usuario.

Este portal se encargará de las siguientes funciones:

- El envío de un formulario con las opciones de análisis seleccionadas por el usuario y el archivo de la partitura a analizar.
- La extracción de datos del archivo XML de la partitura y escritura de los mismos en ficheros según se describe en la sección anterior.
- La ejecución de los módulos seleccionados del sistema experto.
- Mostrar los errores detectados.

5.2.1. Tratamiento del archivo XML

Como se ha comentado antes, CLIPS no puede extraer la información de las etiquetas del archivo XML. Esta tarea es implementada por Python haciendo uso de la API *ElementTree XML*. Esta API nos permite buscar etiquetas y leer el contenido de las mismas.

Los archivos XML que contienen partituras, hacen uso de una gran cantidad de etiquetas. En este caso, para extraer los datos necesarios consultamos las siguientes:

- *Key*: para extraer la tonalidad.
- *Time*: para extraer el compás.
- *Note*: para extraer información de la nota. Dentro de ésta consultamos la siguientes subetiquetas:
 - *Pitch*: para extraer el nombre y la octava.
 - *Accidental*: para comprobar si tenía una alteración accidental.
 - *Type* y *Dot*: para extraer la duración.
 - *Voice*: para extraer la voz que canta dicha nota.

La información contenida en las etiquetas se escribe en ficheros de acuerdo al formato que definimos.

Determinación de movimientos melódicos

Para el análisis melódico y algunas cuestiones del análisis armónico es necesario conocer la dirección que van tomando las melodías. Esta información no viene representada de forma explícita en el archivo de la partitura, sino que debemos extraerla nosotros.

Una vez se han creado los archivos que contienen las notas de las voces, vamos comprobando las notas dos a dos para determinar si la melodía hace un movimiento descendente, ascendente o se mantiene.

La lista de movimientos de cada voz se escribe en un fichero para ser utilizado por los módulos de análisis.

5.2.2. Integración del sistema experto

Esta fase de la implementación fue la que presentó más problemas. Para integrar el sistema con la interfaz web he utilizado la librería PyCLIPS. Esta librería permite utilizar las funciones de CLIPS para crear sistema expertos en Python. Además, también permite cargar archivos creados en CLIPS y ejecutarlos.

Gracias a esta última característica, en un principio se pensó que únicamente sería necesario cargar los archivos de los módulos y ejecutarlos cuando fuera necesario. Sin embargo, hubo que hacer algunas modificaciones.

El primer problema con el que me encontré fue que los archivos no podían cargarse. Tras buscar en la documentación de PyCLIPS [Garosi, 2008], encontré la respuesta a este problema. PyCLIPS permite cargar archivos de CLIPS que contengan reglas o hechos, pero no ambas en un mismo fichero. Así pues, dividí los ficheros en ficheros con reglas y ficheros con hechos.

De nuevo, surgió otro problema, esta vez con los archivos de hechos. En este caso, no se aceptaban las definiciones de las estructuras `deftemplate`. Consulté de nuevo la documentación y también varios foros, sin encontrar demasiada información al respecto ni un motivo real por el cual no estuviera funcionando. Tras consultar también la documentación propia de CLIPS [Riley, 2015b, Riley, 2015a] llegué a la conclusión de que los `deftemplate` no podían ser aceptados como hechos. Así que para solucionarlo, decidí añadirlas utilizando explícitamente las funciones de PyCLIPS. A

pesar de que parecía haberse solucionado, esto derivó en nuevos problemas asociados a dos funciones de CLIPS: las funciones `clear` y `reset`.

La función `clear` borra todo el entorno de CLIPS, eliminando de la memoria de trabajo todos los hechos, reglas, plantillas y funciones que estuviesen definidos.

La función `reset` reinicia el entorno, eliminando sólo los hechos y dejando en la memoria de trabajo todas las reglas, plantillas y funciones.

Antes de la ejecución de cada módulo se siguen los siguientes pasos:

- Se transforma el XML a los archivos de entrada del sistema experto.
- Se limpia el entorno.
- Se reinicia el entorno.
- Se cargan las plantillas del módulo.
- Se cargan los hechos del módulo.
- Se cargan las reglas del módulo.

Tras consultar con algunos usuarios de la comunidad y preguntar en varios foros llegué a la conclusión de que debido a algún problema con la librería PyCLIPS o el intérprete de Python, la función `clear` no puede ejecutarse correctamente. Esto provoca que cada vez que se lancen los módulos del sistema, se redefinan las plantillas con las funciones de PyCLIPS. CLIPS no permite sobrescribir hechos o plantillas, por lo que nos daba un error de redefinición.

Para el correcto funcionamiento del sistema, las plantillas debían ser definidas una única vez al inicio del sistema. Tras consultar la documentación de Django y el foro de la comunidad encontré una solución: había que definir las plantillas en el archivo `urls.py`. Este archivo contiene las expresiones regulares de las “urls” en las que se visualizarán las secciones de la pagina web. Éste código se ejecuta una única vez al inicio de la aplicación, lo que nos permite evitar que las plantillas se creen una y otra vez.

Aunque se solucionó el problema de las plantillas, no se podía leer el archivo de hechos. Para asertar conjuntos de hechos en el entorno de CLIPS se utiliza la instrucción `deffacts`. Esta instrucción no era reconocida por la función `LoadFacts` de PyCLIPS, que es la encargada de cargar el archivo. Intenté modificar la sintaxis del archivo utilizando otras instrucciones como `assert`, que “aserta” los hechos en la

memoria de trabajo, o escribiendo directamente los hechos, sin éxito. Al igual que con la función `clear`, no conseguía hacer que la función `LoadFacts` se ejecutase correctamente. De nuevo, consulté a la comunidad y busqué en la documentación pero no encontré ninguna respuesta. También busqué otros ejemplos de sistemas expertos que se hubiesen implementado utilizando estas tecnologías y, pese a encontrarlos, comprobé que utilizaban las mismas funciones y recursos que estaba empleando yo.

Finalmente, opté por almacenar el contenido de los hechos en archivos independientes - un archivo por tipo de `template` - y añadir nuevas reglas con máxima prioridad para la lectura de estos ficheros y la creación de los hechos correspondientes. De esta manera, los hechos de la base de conocimiento se asertan sin problemas al inicio del sistema.

Algunos de estos problemas referidos a funciones e instrucciones que no se ejecutan o dan errores de ejecución se deben principalmente a que PyCLIPS define los hechos, reglas y demás elementos de CLIPS como objetos. Esto hace que se den problemas de asignación y liberación de memoria, como en el caso de la instrucción `clear`, entre otros.

También comprobé que el uso de PyCLIPS mediante comandos en la consola no daba ninguno de estos problemas. Esto sugiere que quizá haya algún problema con el intérprete a la hora de ejecutar scripts que contengan código de esta librería.

En cualquier caso, una vez se solucionaron estos problemas, se pudo completar la integración del sistema de manera satisfactoria.

5.2.3. Visualización de errores

Se contemplaron varias posibilidades para mostrar los errores:

- Mostrar los errores en tiempo real según fuesen encontrados por el sistema.
- Mostrar las listas completas de errores tras finalizar todo el proceso de análisis.
- Mostrar gráficamente la sección de la partitura en la que se detecte un error, señalizando de alguna manera las notas implicadas.

De estas opciones, se decidió que la segunda cumplía mejor con los requisitos de los usuarios. Esto se debe al hecho de que los errores que detecta el sistema, aunque se encuentren localizados en un compás concreto de la partitura, el ámbito al que afecta el error no es local sino global. Esto quiere decir que un error puede afectar

al resto de la partitura al ser corregido generando nuevos errores, agravando otros o incluso solucionando otros problemas posteriores.

Por este motivo, mostrar los errores en tiempo real no le resulta tan útil al usuario como mostrar la lista completa. El tener la lista completa permite tener una visión general de todas las faltas y errores.

De igual manera, el hecho de mostrarlo gráficamente no resultaba especialmente útil por la posibilidad de que existan secciones que se repitan o se parezcan a lo largo de la obra, dificultando la señalización concreta de los errores.

En todos los errores encontrados se especifica el compás, la parte, el tipo de error encontrado, las voces involucradas y una explicación sobre el mismo, haciendo alusión al porqué es producido y la razón por la que no debe darse. Si no se encontrasen errores, el sistema mostraría un mensaje de éxito.

5.3. Aspecto final

En las siguientes figuras podemos ver cuál ha sido el aspecto final de este sistema:


The screenshot shows the main interface of a web application titled "Sistema Experto de Análisis Musical". The interface has a dark blue header and footer. The main content area has a red background. In the center, there is a white rounded rectangle containing the following elements:

- The text "Selecciona de las siguientes opciones lo que quieres analizar:" followed by three radio button options:
 - ☒ Módulo 1: quintas, octavas, tritonos y duplicación de sensibles
 - ☐ Módulo 2: fallos melódicos y de contrapunto
 - ☐ Módulo 3: análisis armónico, búsqueda de acordes incompletos
- The text "Selecciona la partitura:" followed by a file selection button labeled "Seleccionar archivo" and the text "partitura.xml".
- An "Enviar" button on the right side of the white box.

At the bottom of the footer, it says "Made with ♥".

Figura 5.3.1: Página principal

Sistema Experto de Análisis Musical

 Por favor, espere mientras se procesa su partitura

Selecciona de las siguientes opciones lo que quieres analizar:

- ☒ Módulo 1: quintas, octavas, tritonos y duplicación de sensibles
- ☐ Módulo 2: fallos melódicos y de contrapunto
- ☐ Módulo 3: análisis armónico, búsqueda de acordes incompletos

Selecciona la partitura:

partitura.xml

Figura 5.3.2: Página principal tras enviar el formulario

Sistema Experto de Análisis Musical

Resultados

Errores Módulo 1

En la 4ª parte del compás 8 las voces contraalto y tenor duplican la sensible de la tonalidad. La sensible es una nota que provoca inestabilidad y que se siente atraída por resolver en la tónica de la tonalidad. Al duplicar la sensible generaremos octavas paralelas, lo cual es una falta que no se puede dar en este estilo de obras.

En la 4ª parte del compás 6 las voces soprano y contraalto duplican la sensible de la tonalidad. La sensible es una nota que provoca inestabilidad y que se siente atraída por resolver en la tónica de la tonalidad. Al duplicar la sensible generaremos octavas paralelas, lo cual es una falta que no se puede dar en este estilo de obras.

En la 3ª parte del compás 6 las voces soprano y contraalto duplican la sensible de la tonalidad. La sensible es una nota que provoca inestabilidad y que se siente atraída por resolver en la tónica de la tonalidad. Al duplicar la sensible generaremos octavas paralelas, lo cual es una falta que no se puede dar en este estilo de obras.

Made with ♥

Figura 5.3.3: Visualización de resultados con errores



Figura 5.3.4: Visualización de resultados sin errores

6. Pruebas y validación

Con toda la implementación del sistema realizada, pasamos a la fase de verificación y validación del sistema.

6.1. Verificación

Como describimos en el capítulo anterior, la verificación del sistema experto consistió en comprobar que accedía y leía bien el contenido de los ficheros y añadía los hechos a la memoria de trabajo de manera correcta.

Como no podíamos comprobarlo visualmente como hicimos con el prototipo, se hicieron pruebas en las que el sistema experto, además de los errores, mostraba los hechos asertados, hechos calculados y hechos leídos.

Una vez se comprobó que el sistema experto se ejecutaba correctamente, pasamos a comprobar las funcionalidades de la interfaz.

Primero se comprobó que las funciones de extracción de información del archivo XML obtenían y guardaban correctamente los datos. Para ello, se ejecutaron todas las funciones con distintas partituras y se revisaron los ficheros resultantes para ver si coincidían con el formato escogido y no contenían fallos.

También se verificó el funcionamiento del envío del formulario para que no se produjesen las siguientes situaciones:

- No se puede enviar un formulario en blanco.
- No se puede enviar un archivo sin seleccionar ninguna opción.
- No se puede enviar la lista de opciones sin adjuntar un archivo.
- El archivo a adjuntar debe ser de formato XML.

Por último, se examinó que los resultados se mostrasen de manera legible y ordenados en sus pestañas correspondientes. Las pestañas de los módulos deben mostrarse si se ha seleccionado dicho módulo y si se han obtenido errores. En caso contrario, no deberían mostrarse.

6.2. Validación

La validación del sistema experto fue llevada a cabo por Clara Luz Fernández Vecino, profesora de armonía del Conservatorio Ángel Barrios de Granada.

Para ello, el sistema analizó ejercicios realizados por los alumnos del conservatorio:

- Alrededor de 100 ejercicios de armonía. Con estos ejercicios se validaron los módulos de errores provocados por los movimientos y algunos errores armónicos.
- 20 ejercicios de composición de corales. Con estas composiciones se validaron todos los módulos, especialmente el módulo de análisis melódico.

Tras ejecutar el sistema, Clara Luz revisó los resultados obtenidos por el sistema comparándolos con las correcciones que ella misma había llevado a cabo sobre todos los ejercicios.

7. Resultados y conclusiones

7.1. Resultados

Los resultados obtenidos tras la validación del sistema por Clara Luz han sido entre un 85-90 % de acierto. Esto significa que de todos los errores señalados por Clara Luz, el sistema detectó el 85-90 % de todos ellos aproximadamente.

En los casos de errores no detectados, se debía a que eran situaciones extremas o excepcionales que no había considerado para el desarrollo del sistema, como por ejemplo resoluciones excepcionales de séptimas o detección de quintas paralelas entre acordes sin tener en cuenta las notas de paso.

Sin embargo, el sistema en muchos de los ejemplos, fue capaz de detectar errores que no habían sido encontrados por Clara Luz. Esta cantidad de errores podían ser desde encontrar un error no señalado a incluso diez o más, por lo que es difícil estimar un porcentaje claro.

Si no tenemos en cuenta el porcentaje de error debido a casos no contemplados por el sistema o no implementados, el sistema encuentra prácticamente todos los errores, suponiendo un 95 % o más de acierto.

Respecto al tiempo empleado para analizar las partituras, si comparamos el tiempo empleado por el sistema con la interfaz web y sin ella, los resultados no son muy favorables. La ejecución del sistema experto en el entorno de CLIPS tarda unos pocos segundos, mientras que con la interfaz tarda entre 1 y 5 minutos.

7.2. Conclusiones

En relación a los resultados referidos al porcentaje de errores encontrados, considero que dada la complejidad de la tarea, se han obtenido buenos resultados.

He de destacar el hecho de que el sistema encontró errores que no habían sido señalados por Clara Luz. Esto supone que el sistema, en algunas ocasiones, puede analizar partituras mejor o con más detalle que un experto. Obviamente en el caso del experto siempre está el factor humano y la garantía de cometer errores más fácilmente que una máquina. Aún así, este hecho fue una muy grata sorpresa.

También destacar el hecho de que si en la partitura se encuentra alguno de los errores o faltas que se describen en los requisitos, siempre es capaz de detectarlos y señalarlos, lo cual satisface por completo los objetivos propuestos.

En cuanto a los resultados sobre el tiempo de ejecución, el hecho de crear la interfaz web ha supuesto una pérdida de velocidad considerable.

Al haber embebido el sistema dentro de una aplicación web, se ha perdido una de las mejores características de CLIPS, que es su velocidad de procesamiento. Esta ventaja de CLIPS se ha sacrificado para poder hacer un sistema más accesible y con una interfaz más afable, dado que es muy improbable que los usuarios finales tengan algún conocimiento sobre CLIPS o su existencia.

Por otro lado, para un experto, estos análisis tan exhaustivos pueden llevar alrededor de 20 minutos para un coral de 24-32 compases, que suele ser la longitud media de este tipo de obras. Teniendo en cuenta esta comparación, el sistema tarda, como mínimo, cuatro veces menos que un experto, lo cual supone una ganancia en tiempo bastante aceptable.

Dado que el tiempo no era realmente un requisito indispensable o un objetivo propuesto no le he dado demasiada importancia a la pérdida de eficiencia que nos ha supuesto la interfaz, y más si tenemos en cuenta que para el usuario supone una mejora respecto a hacer el análisis manualmente.

Aún así he realizado algunas pruebas y mediciones sobre la ejecución del sistema con la interfaz. He podido comprobar que la pérdida de velocidad no radica en la ejecución del sistema experto sino que parece estar más bien relacionada con procesar los resultados para poder mostrarlos de manera correcta. Esto significa que el problema no está relacionado con la implementación del sistema experto sino con alguno de los procedimientos que se llevan a cabo en la interfaz o con su conexión entre el servidor y el usuario.

Teniendo en cuenta todo lo anterior, podemos concluir que este sistema proporciona unos resultados bastante buenos en un tiempo bastante aceptable y que hemos podido satisfacer todos los requisitos y alcanzar los objetivos propuestos.

7.3. Futuro desarrollo

Uno de los motivos principales por los que se escogió hacer un sistema experto es por su gran escalabilidad. Este proyecto puede ser ampliado en una gran cantidad

formas:

- Añadiendo nuevas reglas de armonía y/o excepciones a los módulos de análisis ya existentes.
- Añadiendo nuevos módulos para analizar otros aspectos o estilos musicales.
- Adaptando los módulos a otros tipos de obras y formas musicales, como obras para piano, orquesta, ...
- Añadir nuevas funcionalidades a la aplicación como creación de usuarios, llevar un registro de las partituras analizadas, añadir estadísticas, ...

Además, está el hecho de que se puede intentar mejorar la eficiencia del sistema utilizando programación concurrente, mejorando el uso de la memoria o incluso intentar trasladarlo a otro tipo de interfaz que pueda ser más eficiente en términos de velocidad.

En definitiva, este sistema aún tiene muchas posibilidades de mejorar y crecer, aunque como primera versión ha dado unos resultados más que satisfactorios.

Anexo: Glosario

Acorde: conjunto de tres o más notas diferentes que suenan simultáneamente y que constituyen una unidad armónica. Las notas que forman los acordes de triada o tres notas se denominan: *fundamental* - nota sobre la que se construye el acorde -, *tercera* - nota que forma un intervalo de tercera con la fundamental - y *quinta* - nota que forma un intervalo de quinta con la fundamental. En el caso de acordes de cuatriada o cuatro notas, además de las notas que se mencionan, aparece la *séptima* - nota que forma un intervalo de séptima con la fundamental.

- **Acorde invertido:** se dice que un acorde está *invertido* cuando la nota más grave no es la fundamental, sino alguna de las otras que forma el acorde. Si se forma sobre la tercera, el acorde está en *primera inversión*; si se construye sobre la quinta, está en *segunda inversión* y si se construye sobre la séptima, está en *tercera inversión*.

Alteraciones: signos que modifican la entonación o altura de los sonidos naturales y alterados. Las alteraciones más utilizadas son el *sostenido* # , el *bemol* b y el *becuadro* bb.

- **Alteraciones propias:** aquellas que se colocan al principio del pentagrama. Éstas son determinadas por la *tonalidad* de la obra. También se les llama *armadura*.
- **Alteraciones accidentales:** aquellas que aparecen en cualquier punto de la partitura, modificando temporalmente a la nota que acompañan.

Análisis armónico: estudio, a partir de la tonalidad de la obra, de los *acordes*, *grados*, *inversiones* y *funciones* que componen a la misma y las relaciones entre ellos.

Análisis melódico: estudio de la melodía o línea melódica para determinar si está bien construida, es decir, si es fluida y consistente con la *armonía* sobre la que se basa.

Armadura: conjunto de *alteraciones* propias de una tonalidad escritas al principio de un pentagrama, entre la *clave* y el *compás*. Determina qué notas deben ser interpretadas de manera sistemática un semitono por encima o por debajo de sus notas naturales (no alteradas) equivalentes.

Armonía: disciplina de la música en la que se estudian los acordes y sus relaciones para producir sensaciones sonoras, sensaciones de relajación o sosiego (consonancia) y de tensión (disonancia).

- **Armonía clásica o armonía tradicional:** referida a la armonía tonal o funcional. La armonía tonal es la empleada desde la época Prebarroca (siglo XV) hasta el Romanticismo (siglo XIX).

Coherencia musical: decimos que una obra o melodía es coherente cuando ésta tiene fluidez y es consistente.

Conducción melódica: referida a la dirección y *movimientos* entre las notas de una melodía.

Consonancia: percepción subjetiva según la cual se consideran ciertos *intervalos* musicales menos tensos que otros. Conjunto de sonidos que el oído percibe de forma distendida, agradable.

Contrapunto: técnica de composición musical que evalúa la relación existente entre dos o más voces independientes con la finalidad de obtener cierto equilibrio armónico.







Compás: entidad métrica musical compuesta por varias unidades de tiempo (*figuras musicales*) que se organizan en grupos. Se especifica cuántos *pulsos* o partes hay en cada compás y qué figura musical define un pulso.

Coral: obra para canto al unísono (una sola melodía) o polifónico (varias melodías).

Disonancia: percepción subjetiva según la cual se consideran ciertos *intervalos* musicales más tensos que otros. Conjunto de sonidos que el oído percibe de forma desagradable o tiende a rechazar.

Escala: conjunto de sonidos ordenados, notas de un entorno sonoro particular, de manera simple y esquemática. Estos sonidos puede estar dispuestos de forma ascendente (de grave a agudo) o de forma descendente (de agudo a grave), uno a uno en posiciones específicas dentro de la escala, llamadas *grados*.

Estructura armónica: referido al conjunto de *acordes* que conforman una obra.

Figura musical: signo que representa gráficamente la duración de una nota. Las figuras más conocidas son la redonda , blanca , negra , corchea , semicorchea  y fusa .

Forma musical: referida a la estructura de la obra o tipo. Ejemplos: coral, sonata, sinfonía, canción, ...

Función tonal o función armónica: referida al grado de reposo o de inestabilidad (tensión) que aportan los *grados* de una tonalidad. Las funciones tonales son *tónica* (pasajes en reposo), *subdominante* (pasajes intermedios entre reposo e inestabilidad) y *dominante* (pasajes inestables).

Grado tonal: referido a la posición de cada nota dentro de una *escala*. Se designan mediante números romanos correlativos I, II, III, IV, V, VI y VII. Cada grado tiene asociada una *función tonal*:

- I: función de *tónica*.
- II: función de *subdominante*.
- III: función de *tónica*.
- IV: función de *subdominante*.
- V: función de *dominante*.
- VI: función de *tónica*.
- VII: función de *dominante*.

Intervalo: diferencia de altura entre dos notas, medida cuantitativamente (número) en *grados* o notas naturales y cualitativamente (especie) en tonos y semitonos. Cuantitativamente pueden ser segundas, terceras, cuartas, quintas, sextas, séptimas u octavas. Cualitativamente puede ser mayores, menores, justos, aumentados o disminuidos.

Movimientos: referido a los movimientos que pueden darse entre dos notas. Los movimientos pueden ser a nivel melódico o armónico.

- **Movimientos melódicos:** movimientos que se produce al pasar de una nota a otra en una melodía. Puede ser un movimiento por grado conjunto - nos movemos a la nota continua ascendente o descendente - o un movimiento por salto - nos movemos a una nota no continua.

- **Movimientos armónicos:** movimientos que se producen en las notas -dos a dos- al pasar de un acorde a otro. Pueden ser movimientos *directos* (ambas notas se mueven en la misma dirección) , movimientos *paralelos* (ambas notas se mueven en la misma dirección y manteniendo el mismo intervalo entre ellas) , movimientos *contrarios* (ambas notas se mueven en direcciones contrarias) y movimientos *oblicuos* (una nota se mantiene estática y la otra se mueve ascendente o descendentemente).

Pulso: unidad básica que se emplea para medir el tiempo.

Sensible: referida a la séptima nota de la escala que se encuentra a un semitono de la tónica de la tonalidad. También puede hacer referencia al VII grado de la escala, al acorde que se forma sobre dicho grado o a la función tonal que desempeña (dominante).

Séptima de dominante: referida a la séptima del acorde que se forma sobre el acorde de *dominante* de la *tónica*.

Tonalidad: conjunto de sonidos que están en íntima relación entre sí con respecto a un centro tonal o *tónica*.

Tónica: primer *grado* de la *escala* musical y es la nota que define la *tonalidad*.

Bibliografía

- [de Pedro, 1992] de Pedro, D. (1992). *Teoría Completa de la Música*, volume I. Real Musical.
- [de Pedro, 2000] de Pedro, D. (2000). *Teoría Completa de la Música*, volume II. Real Musical.
- [DFS, 2017] DFS (2017). [DJango Documentation](#).
- [Fubini, 2010] Fubini, E. (2010). *La estética musical desde la Antigüedad hasta el siglo XX*. Alianza Música.
- [Garosi, 2008] Garosi, F. (2008). [PyCLIPS Manual](#).
- [Garrido, 2017] Garrido, E. (2017). [Revista Melómanos](#).
- [Grout and Palisca, 1996a] Grout, D. J. and Palisca, C. V. (1996a). *Historia de la música occidental*, volume I. Alianza Música.
- [Grout and Palisca, 1996b] Grout, D. J. and Palisca, C. V. (1996b). *Historia de la música occidental*, volume II. Alianza Música.
- [Henckmann and Lotter, 1992] Henckmann, W. and Lotter, K. (1992). *Diccionario de estética*. CRÍTICA.
- [Llovich, 2013] Llovich, J. S. (2013). [Quintas y octavas prohibidas en el periodo modal-tonal](#).
- [López, 2017] López, L. O. T. (2017). [Código de la aplicación](#).
- [Michels, 1999] Michels, U. (1999). *Atlas de música*. Alianza Editorial.
- [Otto and Thornton, 2017] Otto, M. and Thornton, J. (2017). [Bootstrap References](#).
- [Peña, 2016] Peña, J. L. C. (2016). Apuntes de la asignatura ingeniería del conocimiento.
- [Piston, 2007] Piston, W. (2007). *Armonía*. Mundimúsica Ediciones.
- [Resig, 2017] Resig, J. (2017). [jQuery Documentation](#).
- [Riley, 2015a] Riley, G. (2015a). [CLIPS Reference Manual. Volume II. Advanced Programming Guide](#).

- [Riley, 2015b] Riley, G. (2015b). [CLIPS Reference Manual. Volume I. Basic Programming Guide.](#)
- [Ronacher, 2017] Ronacher, A. (2017). [Jinja2 Documentation.](#)
- [Schoenberg, 2000] Schoenberg, A. (2000). *Fundamentos de la composición musical.* Real Musical.
- [van Rossum, 2017] van Rossum, G. (2017). [Python Documentation.](#)
- [WHATWG, 2017] WHATWG (2017). [HTML5 Documentation.](#)
- [Zamacois, 2002a] Zamacois, J. (2002a). *Tratado de Armonía*, volume I. Idea Musica.
- [Zamacois, 2002b] Zamacois, J. (2002b). *Tratado de Armonía*, volume II. Idea Musica.