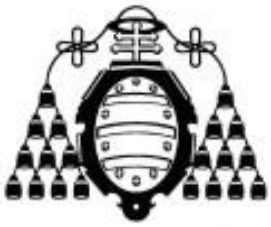
UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

PROYECTO FIN DE CARRERA

"Herramienta de programación para formas de vida artificiales basada en DSLs: PleoProg"

DIRECTORES: **Oscar Sanjuán Martínez**

B. Cristina Pelayo García-Bustelo

AUTOR: **David García Garmendia**

Vº Bº del Director del Proyecto

Agradecimientos

Agradezco todo el apoyo que me han dado durante toda la carrera, y especialmente durante el desarrollo de este proyecto, a mi madre Mayte y mi novia Alicia.

Agradezco también la ayuda recibida por el creador de YAPT y webmaster de Aibopet que se invirtió parte de su tiempo en explicarme algunos fundamentos básicos para iniciar el desarrollo de este proyecto.

Resumen

Este proyecto ha servido para crear una herramienta que permita la programación rápida, sencilla e intuitiva de la forma de vida artificial Pleo. Tiene como principal objetivo que personas sin ningún tipo de conocimientos de programación y/o robótica sean capaces de crear un diagrama que represente una personalidad a seguir por la forma de vida artificial mediante la aplicación de escritorio desarrollada en este proyecto.

Básicamente consiste en la creación de una aplicación que traduzca los diagramas creados en código ejecutable por el robot, como si de un compilador se tratase. Estos diagramas realizados por el usuario, junto con el primer archivo XML generado, consolidan un DSL que sirve de entrada para el subsistema compilador encargado de traducirlo a un programa ejecutable, inicialmente, por la forma de vida artificial, Pleo.

Pleo es un robot que puede ser empleado para diversos usos, y esta herramienta amplia sus utilidades. La más evidente es la relativa al juego y las posibilidades que ofrece a los niños para disfrutar de este robot, ya que el robot tiene su propia personalidad y no es posible decirle que hacer, es necesario tener ciertos conocimientos para crear pequeños programas y darle órdenes concretas, es por esta misma razón que también tiene utilidad en el ámbito docente pues permite un primer acercamiento a la programación con resultados visibles y amenos.

Para desarrollar el proyecto se hace uso del PDK (Pleo Development Kit) para obtener programas ejecutables por Pleo. Este kit fue puesto a disposición de los desarrolladores por la propia empresa fabricante y es parte indispensable para la creación de los programas. Este kit además incluye otros muchos recursos que ayudan a enriquecer los programas creados.

Palabras Clave

- Robótica
- DSL
- Aprendizaje
- PDK
- XML
- C#

Abstract

This project has served to create a programming tool that allows programming quick, easy and intuitive the artificial life form, Pleo. Its main objective is that people without any programming and / or robotics knowledge be able to create a diagram that represents a character to follow by the artificial life form with the desktop application developed in this project.
It basically consists on creating an application that translates diagrams into executable code for the life form. These diagrams made by the user generated a XML file that is a DSL. This XML is the input for the compiler subsystem responsible for translating it into an executable program, initially, by the artificial life form, Pleo.

Pleo is a robot that can be used for different applications and this increases its profits. The most obvious is the playful application, specially for the children, but not only. The robot has its own personality and you can not tell what to do, you must have some knowledge to create small programs and give commands. For this reason the tool it is useful for teaching because it allows a first approach to programming with visible results and fun.

To develop the Project, the tool makes use of the PDK (Pleo Development Kit) for make executable programs for Pleo. It also was made available to developers for Pleo's owner company and is an indispensable part for the creation of programs. This kit also includes many other resources that help enrich the programs created.

Keywords

- Robotics
- DSL
- Learning
- PDK
- XML
- C#

Índice General

Índice de Figuras

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

Pleo es una forma de vida artificial, en este caso un robot, diseñada para el entretenimiento y el aprendizaje principalmente, con distintas personalidades descargables y diferentes reacciones dependiendo de los estímulos exteriores. Pleo es un auténtico robot pero que no es capaz de acatar órdenes directas naturalmente. Programar una personalidad es difícil siguiendo la documentación aportada por el fabricante, y aunque existen algunas aplicaciones de escritorio que permiten su programación simplificando el proceso, dichas aplicaciones siguen siendo demasiado complejas y dominarlas lleva cierto tiempo. En muchas ocasiones, al intuir un periodo de aprendizaje algo extenso el usuario se desanima y desiste; por norma general solo los usuarios con ciertos conocimientos avanzados consiguen su propósito. Después de considerar los puntos anteriores que presentan las aplicaciones existentes se decidió llevar a cabo el presente proyecto.

Los objetivos del proyecto son elaborar una herramienta para diseñar una personalidad, que dicha herramienta sea sencilla y que cualquier persona sin conocimientos de programación sea capaz de desenvolverse adecuadamente con esta. Además se posibilitará la futura compatibilidad con otras versiones de Pleo, u otros robots, usando la misma herramienta y realizando ninguna o pocas modificaciones de esta. Para ello se dividirá la herramienta en tres subsistemas: la interfaz, que es la que muestra las herramientas y el entorno de trabajo permitiendo crear los diagramas, el gestor de personalidad que se encargará de crear el XML para el compilador y el compilador para traducir el fichero generado por este DSL en un programa que pueda ejecutar el robot.

El alcance es crear una herramienta que cumpla con los anteriores objetivos ofreciendo el mayor número de acciones explotando la mayoría de posibilidades de Pleo y considerando las posibles futuras ampliaciones de las capacidades de Pleo y adaptando el proyecto para facilitar la compatibilidad con otros robots.

1.2 Resumen de Todos los Aspectos

1.2.1 Planificación

Se estudian los aspectos iniciales del proyecto y se detalla la estimación del coste temporal, económico y de personal necesario para llevar a cabo el proyecto. En esta fase se define también el alcance.

1.2.2 Análisis

Esta fase conlleva la comprensión del problema y de todos los aspectos relacionados con este, se desarrolla una idea inicial del proyecto. Se comprenden los requisitos de este y se definen las especificaciones que los cumplan. Además se analizan los casos de uso.

1.2.3 Diseño

A partir del análisis se responde a la cuestión de cómo desarrollar el proyecto, para ello entran en juego varios tipos de diagramas, realizando todos los necesarios, para explicar el funcionamiento del futuro proyecto. Se realiza una descripción detallada de todos los componentes del sistema.

1.2.4 Implementación

Con toda la información de la fase de diseño se obtienen unas especificaciones y es cuando se realiza la programación del sistema mediante las tecnologías escogidas.

1.2.5 Pruebas

Una vez desarrollado el sistema se han creado unas pruebas para verificar que la herramienta cumple con todos los objetivos y esta está exenta de fallos. Esto posibilita que si se detectan se puedan solucionar.

1.2.6 Manuales

Con la función de sacar el mayor provecho a la herramienta y posibilitar su funcionamiento se proveen los siguientes manuales:

- Instalación: Describe los pasos necesarios para instalar la herramienta.
- Usuario: Describe el funcionamiento de la herramienta y sus posibilidades detallando los pasos a seguir para utilizar correctamente la herramienta.
- Programador: Describe la estructura de la herramienta para posibilitar futuras ampliaciones o mejoras por otros programadores.

1.2.7 Conclusiones y ampliaciones

Las conclusiones de realizar el proyecto se plasmarán en este apartado, lo que incluye hablar sobre la toma de ciertas decisiones y si los resultados son los pensados.

Las ampliaciones son las posibles mejoras que no se han llevado a cabo pero que podrían, en un futuro, realizarse por parte del propio creador u otros programadores.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

El proyecto consiste en el desarrollo de una herramienta que facilite al usuario la creación y edición de personalidades para Pleo. Dicha herramienta está enfocada a cualquier público, ya sea un experto programador o un niño que se adentra en el mundo de la robótica a través del juego.

La mayoría de programas usados para programar este robot son complejas herramientas que requieren conocimientos previos para su utilización; este proyecto creará una herramienta que se podrá utilizar sin ningún tipo de aprendizaje previo, que es uno de los principales problemas que existen entre las herramientas que actualmente están disponibles.

Este proyecto desarrollará una herramienta con la misma potencia que otras existentes pero simplificando al máximo su interfaz haciéndola más accesible a todos los públicos.

Otra de las capacidades de la que se dota este proyecto es su adaptación a otros robots mediante pequeñas modificaciones en parte del código permitiendo exportar una misma secuencia de acciones a diferentes robots con poco esfuerzo. Actualmente no se conoce ninguna herramienta que contemple la adición de nuevas formas de vida artificiales para las que pueda programarse.

Si esta herramienta pretende ser usada por cualquier tipo de público, por niños españoles, por ejemplo, es de esperar que el inglés no sea su punto fuerte y este es otra característica destacable, pues las herramientas actualmente disponibles están en inglés. PleoProg se desarrollará inicialmente para hispanohablantes y se contemplará en un futuro su traducción al inglés.

2.2 Objetivos del Proyecto

- Crear una herramienta de programación para Pleo.
- La herramienta debe recoger el mayor número de órdenes posibles.
- La herramienta debe recoger el mayor número de eventos posibles
- La herramienta creada debe ser sencilla e intuitiva.
- La herramienta debe estar diseñada para que en el futuro sea sencillo ampliarla para exportar programas a otros robots.

2.3 Estudio de la Situación Actual

Hay otras herramientas similares que proveen funciones parecidas al presente proyecto. Se analizan a continuación.

2.3.1 Evaluación de Alternativas

2.3.1.1 Pleo Development Kit

2.3.1.1.1 Descripción

Se trata de un conjunto de herramientas oficiales proporcionadas por el fabricante. Consta de varios compiladores de lenguaje PAWN para cada una de las versiones de Pleo, archivos de recursos de sonidos, animaciones, etc., documentación y ejemplos para programar en PAWN. Es una parte esencial de este proyecto, pues es necesario incluir el compilador para crear los archivos ejecutables de la forma de vida artificial.

2.3.1.1.2 Ventajas

La sintaxis de PAWN es muy similar a C, por lo que no es necesario un gran período de aprendizaje para programadores con experiencia previa en C. Además al tratarse del conjunto de herramientas oficiales, existe una comunidad a la que poder recurrir en caso de un problema. Consta de gran número de librerías y recursos con los que programar a Pleo.

2.3.1.1.3 Inconvenientes

El principal inconveniente es que es necesario tener bastantes conocimientos de programación y en ningún caso está dirigido para personas sin experiencia en programación. No posee una interfaz de usuario o entorno donde se facilite la tarea de programación, es necesario crear los archivos de código en editores de texto y posteriormente compilarlos. También es necesario elegir entre los compiladores que se aportan de acuerdo a la versión de Pleo que se tenga. Se encuentra únicamente en inglés.

2.3.1.2 Pleo Personalities

2.3.1.2.1 Descripción

Desde la empresa fabricante y su página web es posible descargar un pack con diferentes personalidades que se instalan en Pleo. Estas personalidades se muestran en las reacciones y comportamientos de Pleo. También hay personalidades creadas por usuarios que se pueden descargar de otras páginas y foros.

2.3.1.2.2 Ventajas

Rapidez, lo que tarde en descargar e insertar el archivo en la memoria de Pleo. Sencillez, el usuario no necesita ningún tipo de conocimiento para hacer uso de las personalidades, elige la que desea aplicar a su robot y la instala, a partir de ese momento Pleo actuará como dicte su personalidad. Fiabilidad, pues las personalidades que se pueden descargar de la página del fabricante se pueden utilizar sin ningún tipo de riesgo para Pleo.

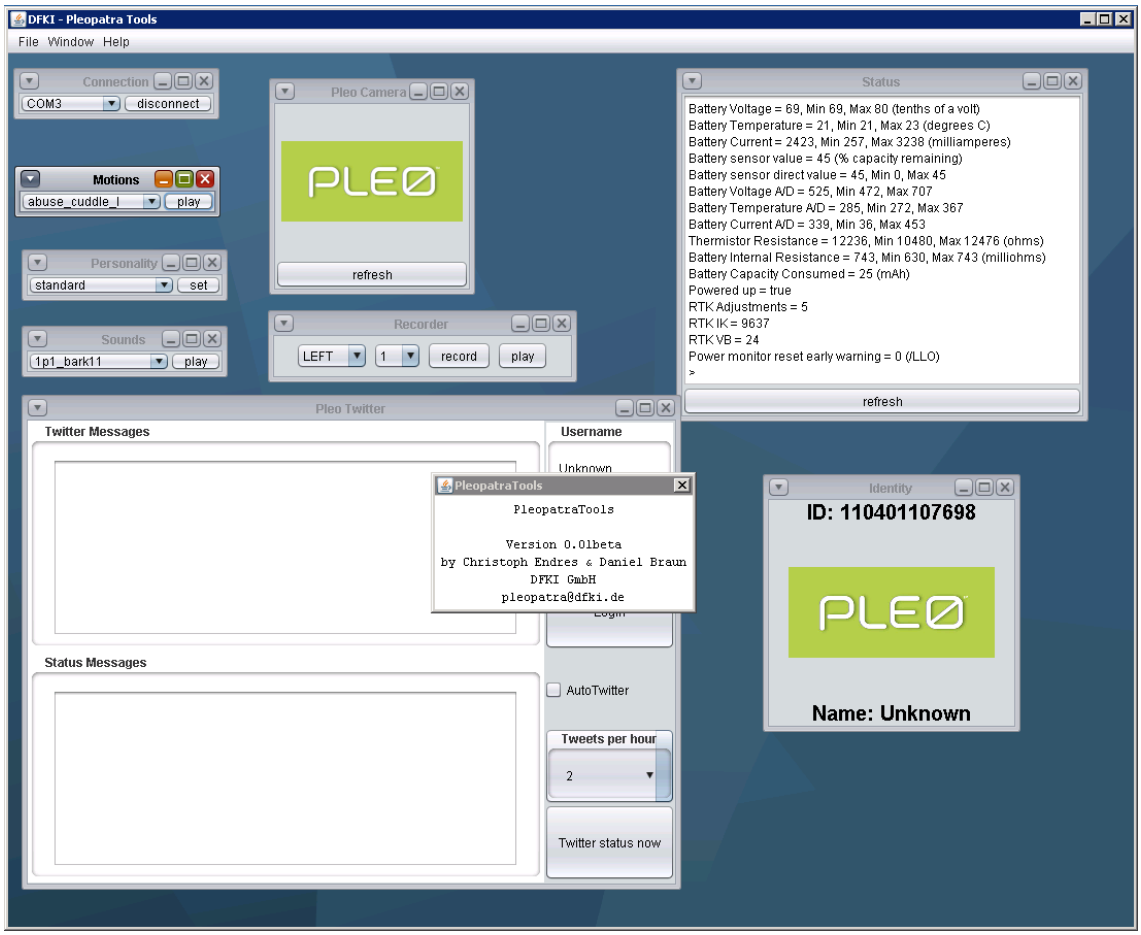
2.3.1.2.3 Inconvenientes

Las personalidades no se puede modificar de una forma sencilla y el usuario no tiene ningún control sobre el robot, no tiene posibilidad de dar órdenes pues es la personalidad instalada quien gobierna a Pleo. El número de personalidades que se pueden descargar es limitado, porque aunque a parte de las oficiales hay otras muchas creadas por los usuarios, no tienen por qué ser lo que deseamos.

2.3.1.3 Pleopatra Tools

2.3.1.3.1 Descripción

Se trata de una herramienta que aprovecha la posibilidad del puerto USB o Serial de Pleo y "Pleo Monitor". Pleopatra proporciona una aplicación con una interfaz amigable que permite realizar todas las operaciones prescindiendo de un terminal.



2.3.1.3.2 Ventajas

Interfaz agradable. Acceso directo y a tiempo real a Pleo. Las órdenes dadas se ejecutan al momento. Estado de Pleo en tiempo real. Facilidad para realizar las acciones con pocos conocimientos.

2.3.1.3.3 Inconvenientes

Su uso está condicionado a que Pleo se encuentre conectado al computador en todo momento. No se puede programar personalidades ni nada similar, solo sirve para órdenes directas. Se encuentra en fase beta y tiene pocas posibilidades. Por la misma razón posee algunos fallos. Se encuentra únicamente en inglés.

2.3.1.4 MySkit

2.3.1.4.1 Descripción

Herramienta que permite la creación de secuencias de movimientos y sonidos mediante una interfaz. Posee una línea de tiempo para el sonido y otra para el movimiento.

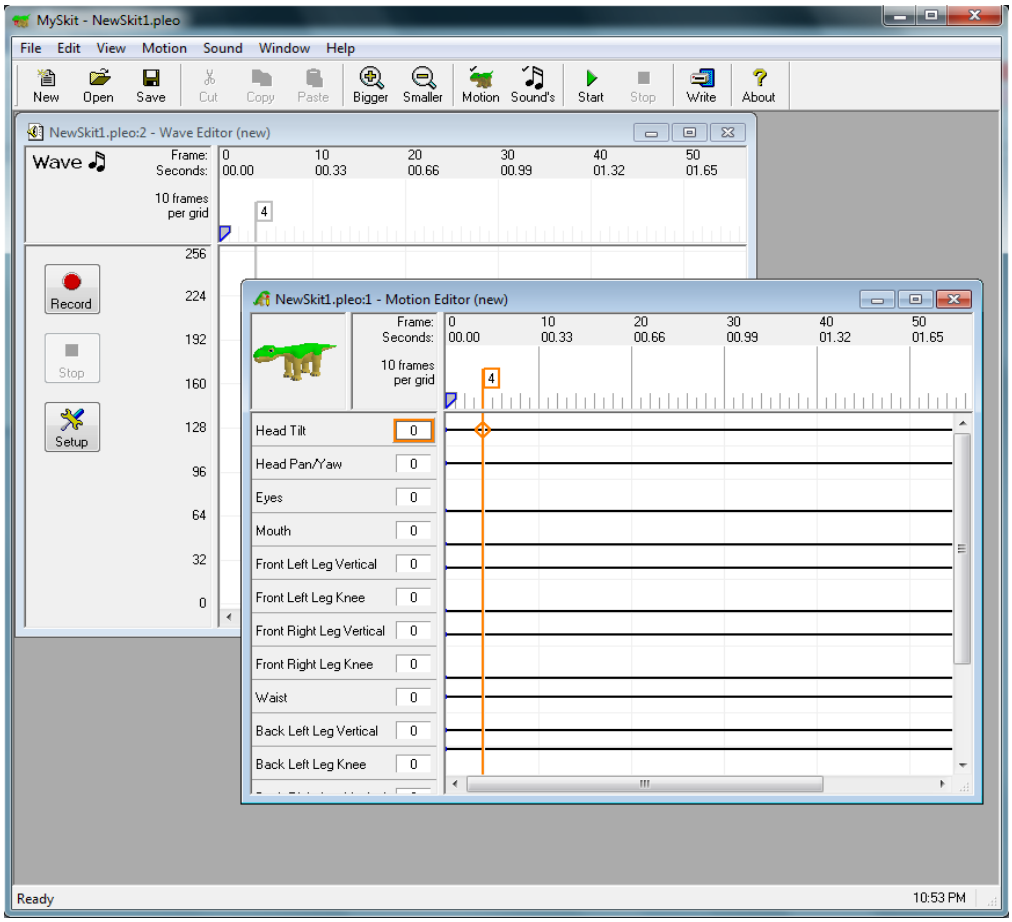


Figura 2.1 MySkit

2.3.1.4.2 Ventajas

Posee una interfaz que facilita la creación y edición de las secuencias de movimientos y sonidos. La herramienta incluye una pequeña aplicación que permite visualizar en un modelo de Pleo virtual una reproducción previa de dicha secuencia.

2.3.1.4.3 Inconvenientes

Interfaz poco intuitiva. Las secuencias están limitadas a los movimientos y sonidos, desperdiciando las posibilidades de Pleo. No es posible crear personalidades que hagan a Pleo comportarse de cierta manera ante los estímulos exteriores. Se encuentra únicamente en inglés.

2.3.1.5 YAPT

2.3.1.5.1 Descripción

Herramienta concebida para editar personalidades existentes. Suele ser utilizada conjuntamente con MySkit para crear secuencias de acciones o comportamientos.

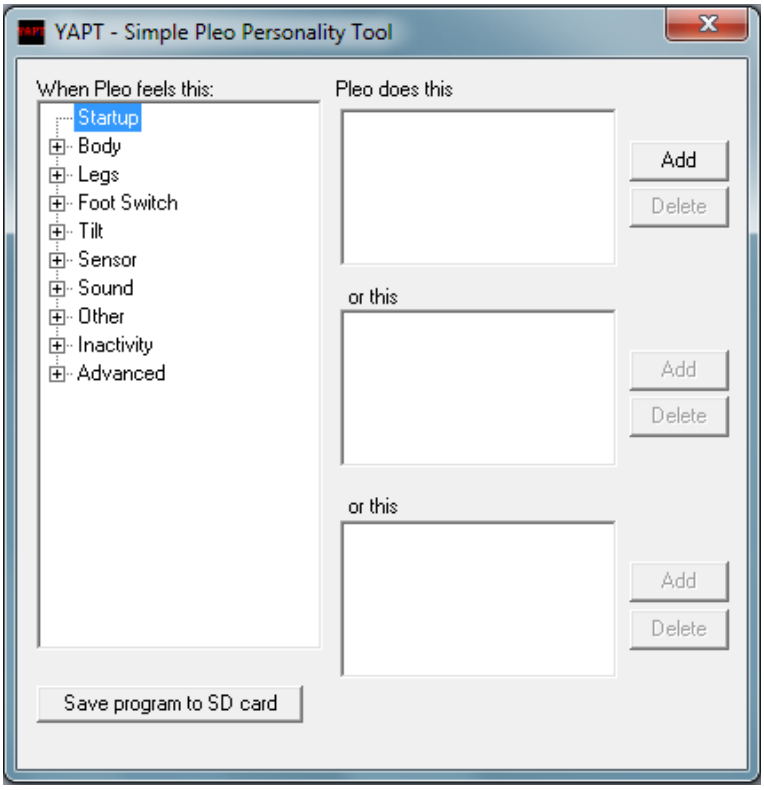


Figura 2.2 YAPT

2.3.1.5.2 Ventajas

Gran cantidad de acciones y sensores disponibles. Incorpora la misma aplicación que MySkit para previsualizar en un modelo como quedan las animaciones sin necesidad de instalar el archivo en el robot.

2.3.1.5.3 Inconvenientes

No crea personalidades, tan solo modifica existentes, por lo que es necesario crear una previamente o importar alguna de las que existen en Internet. Obsoleto y discontinuado ya no recibe ningún tipo de soporte ni actualización. Interfaz poco amigable, no llega a resultar complejo pero es necesario tener algunos conocimientos para poder ponerlo en marcha y comprender el funcionamiento. No es posible adaptarlo para exportar estas personalidades o acciones a otro robot. Se encuentra únicamente en inglés.

Capítulo 3. Aspectos Teóricos

3.1 Conceptos

3.1.1 Robótica / Robots

La robótica es la ciencia que estudia y desarrolla los robots. El término robótica es utilizado por primera vez por Karel Capek en 1920, escritor checo que utilizó esta palabra derivada de *robota* que significa "esclavo" en la novela R.U.R. (Robots Universales Rossum). Sin embargo, la creación de robots se remonta siglos atrás y podemos decir que se refiere a casi cualquier invención, en algunos casos con forma humana o animal, que era creada para descargar el trabajo de los hombres. Actualmente también se desarrollan robots para el ocio y entretenimiento imitando comportamientos animales o humanos.

Figura 3.1 ASIMO

ASIMO (Advanced Step in Innovative Mobility) es uno de los robots que más fama ha tenido debido al hecho de ser uno de los primeros capaces de subir escaleras. Con aspecto humanoide, un peso de 54 Kg. y una altura de 1,30 m es considerado uno de los robots, sino el que más, más avanzados del planeta, en cuanto a movimientos se refiere. El proyecto ASIMO desarrollado por HONDA desde el año 2000 tiene como objetivo reproducir los movimientos humanos para poder aplicarlos a futuros proyectos. ASIMO puede además caminar, correr, saltar, girar y desplazarse lateralmente entre otros muchos movimientos humanos. Se han visto algunas aplicaciones de ASIMO en las que el robot es capaz de jugar al fútbol o dirigir con una batuta una orquesta.

Figura 3.2 TOPIO

En la imagen anterior se muestra a TOPIO (TOSY Ping Pong Playing Robot) creado por TOSY en 2005 y del que han desarrollando varios modelos. Se trata de un robot de forma humanoide capaz de jugar al ping-pong. Tiene un peso de 120 Kg. y una altura de 1,88 m. Dispone de dos cámaras de alta velocidad que le permiten captar con precisión la localización de la pelota y ejecutar los movimientos apropiados para golpearla.

ASIMO y TOPIO son dos ejemplos de la robótica más avanzada existente en la actualidad.

Hoy en día los robots se encuentran en todas partes, desde complejas factorías industriales hasta las cocinas de nuestros hogares donde preparan los alimentos. La aplicación de los robots abarca toda la sociedad pues el fin último es el de sustituir al humano en trabajos o labores. Normalmente se programan los robots para desarrollar una sola tarea durante toda su vida o son manejados a distancia por un operador.

En este proyecto el fin es poder crear aplicaciones para robots, específicamente formas de vida artificiales, pues poseen personalidad que les hace reaccionar de diferente forma a diferentes estímulos exteriores tomando decisiones ellos mismos sin intervención humana, imitando a los animales. El proyecto es una puerta de entrada a la robótica para no expertos o iniciados y constituye una perfecta primera toma de contacto con este tipo de tecnología.

3.1.2 Formas de vida artificial

El primero en utilizar el término vida artificial fue el científico informático Christopher Langton, organizador de "Primera Conferencia Internacional de la Síntesis y Simulación de Sistemas Vivientes" en 1986. El concepto de vida artificial es muy amplio y engloba diferentes aplicaciones, pero básicamente cuando hablamos de formas de vida artificial estamos hablando de sistemas, creados por el hombre, que muestran propiedades de los seres vivos para su estudio, o como en este caso, para el ocio y educación. Es decir, la vida artificial estudia la lógica de los sistemas vivos en entornos artificiales.

Estos sistemas artificiales pueden estar compuestos de diversas formas, pueden ser únicamente software dentro de un computador o tener un hardware específico para desarrollarse e interactuar. Otras posibles formas son las creaciones biológicas en laboratorios.

Tradicionalmente se habla de 3 tipos de formas de vida artificial:

- Soft, de software.
- Hard, de hardware.
- Wet, de bioquímica.

Hay que distinguir el concepto forma de vida artificial y robot pues una forma de vida artificial tiene como objetivo imitar el comportamiento de una forma de vida. Por tanto, ni todos los robots son formas de vida artificiales, ni todas las formas de vida artificiales son robots. En el actual contexto, este concepto se refiere a robots, más concretamente, a robots con personalidad que en la mayoría de los casos son usados para el ocio y entretenimiento (aunque numerosos son los proyectos que los usan en educación, acompañamiento, etc.) Algunos ejemplos son Pleo, Aibo y QRIO.

Figura 3.3 AIBO

Aibo es un robot doméstico con forma de perro creado por Sony en 1999. Tiene sensores para detectar los obstáculos y poder evitarlos. Aibo es el más exitoso de los robots de este tipo que se ha comercializado. En 2003 Sony canceló su comercialización. Su software está diseñado para simular el crecimiento de una forma de vida, que a partir de los estímulos exteriores y dependiendo de cómo su dueño se comporte con este, evolucionará hasta una personalidad u otra que determinará las reacciones que tendrá en el futuro.

Figura 3.4 QRIO

QRIO (Quest for cuRIOsity) es un robot de forma humanoide de 60 cm y 7,3 Kg. desarrollado por Sony del que poco se ha llegado a saber pues nunca se llego a vender, tan solo apareció en algunas ferias tecnológicas. Era más avanzado que Aibo en ciertos aspectos pues estaba dotado de reconocimiento vocal y facial. Además fue el primer robot bípedo capaz de correr. El proyecto fue cancelado el mismo día en el que se cancelo el proyecto Aibo.

Principalmente se ha desarrollado este proyecto para la forma de vida artificial Pleo, del que se detallan sus características más adelante. Futuras ampliaciones de este proyecto podrían implementar soporte para Aibo u otros futuros robots o versiones no comercializados todavía.

Hay que tener en cuenta que el propósito de este proyecto es conseguir que el usuario sea capaz de crear una personalidad totalmente a su gusto para que el robot la ejecute, dejando de un lado las posibles decisiones que el robot pudiera tomar por su cuenta y ejecutando exclusivamente el programa que le instale el usuario.

3.1.3 DSL

El concepto de DSL o Domain-Specific Language (lenguaje específico de dominio) hace referencia a todo lenguaje de programación creado para un problema o situación específico. A diferencia de los lenguajes de propósito general, que deben servir para crear aplicaciones variadas (JAVA, C, PHP, etc.), un DSL está enfocado a una aplicación determinada.

Las ventajas de utilizar un DSL frente a los lenguajes generales es que se ahorra tiempo de programación pues el lenguaje se ajusta mejor a lo que se desea programar y puede ser creado para que sea más sencillo y entendible por otras personas. La contrapartida es que crear un DSL conlleva un coste temporal y aquellos que vayan a utilizarlo necesitaran aprender un nuevo lenguaje.

Algunos ejemplos de DSL son LOGO o R. LOGO es un lenguaje creado específicamente para el ámbito educativo, permite sin tener nociones de programación ir aprendiendo sus fundamentos, es un lenguaje muy simple porque utiliza palabras del lenguaje natural y su programación es muy intuitiva. R es un lenguaje con licencia GPL creado específicamente para trabajar con datos estadísticos, disponible para la mayoría de plataformas y con una gran comunidad que lo respalda

En este proyecto se hace uso de un lenguaje DSL creado específicamente para diseñar la personalidad de los robots. A través de la interfaz el usuario creará el diagrama correspondiente que será grabado en un fichero XML mediante el gestor de personalidades y que a su vez lo enviará al compilador para que lo use para crear los archivos ejecutables por la forma de vida escogida. Este DSL es el principal responsable de que en el futuro se puedan realizar ampliaciones de este proyecto, pues tan sólo será necesario modificar el compilador para que se ajuste a las nuevas formas de vida a las que se desea dar soporte.

3.2 Tecnologías

3.2.1 XML

XML o eXtensible Markup Language (lenguaje de marcas extensible) es un metalenguaje desarrollado por el W3C para almacenar información de forma estructurada. Originalmente IBM creó GML (Generalized Markup Language), utilizado para estructurar documentos técnicos a través de etiquetas. Más tarde pasó a llamarse SGML (Standard Generalized Markup Language) y acogido por ISO en 1986. SGML no es un lenguaje de marcas sino una especificación para la creación de otros lenguajes, un lenguaje derivado de SGML es HTML. HTML tuvo mucho éxito, pero con el tiempo comenzó a mostrar sus limitaciones. La web albergaba documentos de todo tipo y se usaba HTML para cosas muy diferentes para las que se había concebido, los navegadores no eran exigentes con los documentos HTML y estos eran demasiado complejos debido a que tenían que mostrar documentos que en muchas ocasiones contenían errores.

XML surge como una versión simplificada de SGML para describir datos, y al igual que este, no es un lenguaje sino una especificación. Con esta nueva especificación el W3C desarrolló XHTML, que solventaría algunos de los grandes problemas de HTML. Por otra parte XML empezó a usarse en multitud de ámbitos para el almacenaje y manejo de datos por las aplicaciones software ya que facilita el intercambio de datos entre sistemas diferentes.

En este proyecto se usan ficheros XML para almacenar la personalidad creada por la herramienta, estos mismos ficheros XML son la entrada del compilador que generará el programa para la forma de vida artificial Pleo. La ventaja de usar XML es que es independiente de la forma de vida a programar y en un futuro facilitará la ampliación del compilador para que exporte las secuencias a otras formas de vida diferentes a Pleo con una misma secuencia de acciones.

3.2.2 WPF

WPF es una tecnología desarrollada por Microsoft. Inicialmente liberada como parte de .NET Framework 3.0 en 2006.

WPF provee un modelo de programación vista controlador para separar la interfaz de la lógica de negocio, facilitando el trabajo de los diseñadores y programadores que pueden trabajar paralelamente en una misma aplicación. La parte de la interfaz se programa mediante XAML y la lógica de negocio mediante alguno de los lenguajes de .Net. Los gráficos se representan mediante Direct3D proporcionado gráficos más complejos y temas personalizados. Incorpora también un conjunto de servicios para Data Binding (enlace de datos) resultando un método simple para que las aplicaciones trabajen con datos. Otras opciones mejoradas y avanzadas en WPF se usan en animaciones, plantillas, imágenes, documentos, texto, anotaciones y efectos entre otros.

El proyecto se aprovecha de la tecnología WPF frente a otras por la fuerte separación entre la interfaz y la lógica de negocio facilitando la programación y su posterior modificación para futuras ampliaciones. Además WPF ofrece herramientas muy potentes para el diseño de interfaces.

3.2.3 XAML

XAML o eXtensible Application Markup Language (lenguaje extensible de formato para aplicaciones) es el lenguaje de formato para la interfaz para WPF. Es un lenguaje declarativo basado en XML. Su origen es el mismo que el de WPF.

Desarrollado para facilitar la creación de interfaces de usuario que sean independientes de la lógica de negocio de la aplicación. Su uso normalmente está ligado a la programación en la plataforma .Net, y por ello fue diseñado para soportar las clases y métodos de esta plataforma. Los archivos XAML pueden ser creados mediante programas de edición de interfaces como Microsoft Visual Studio o Microsoft Blend, que son capaces de generar dichos archivos automáticamente.

A continuación se muestra, como ejemplo, el trozo de código necesario para declarar un botón visualmente en XAML.

```
<Canvas xmlns="http://schemas.microsoft.com/client/2007"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" >
  <Button Content="Púlsame"/>
</Canvas>
```

La utilización de XAML en estos entornos no es obligatoria, pues todo lo expresado en XAML puede ser programado mediante C#, pero aumentando con ello la complejidad de los programas. Se usa XAML en este proyecto para crear las interfaces de usuario necesarias para la interacción.

3.3 Herramientas

3.3.1 PLEO

Pleo es un robot fabricado por la empresa Ugobe presentando en 2006. En junio de 2009 Jetta Company Limited adquirió los derechos de Pleo aprovechando la bancarrota de Ugobe. Tiene la apariencia de un Camarasaurio de corta edad y desde su nacimiento es capaz de desarrollar una personalidad única gracias a su inteligencia artificial. También es posible descargar diferentes personalidades e instalárselas a través de una tarjeta SD. En el desarrollo de Pleo, Ugobe tomó nota de los sistemas biológicos y neurológicos del Camarasaurus, y re-interpretó estos elementos mediante hardware y software.

El hardware de Pleo consiste en:

- **2 ARM7 CPUs**. 1 en la cabeza para la cámara, micrófono, IRs y los sensores de la cabeza y cuello. 1 en el cuerpo para los sensores, motores y la capa de alto nivel.
- **14 motores**. 2 por pierna, 1 en el torso, 2 en la cola, 2 en el cuello y 1 en la cabeza.
- **8 sensores de tacto**. 1 por pierna, 2 en la espalda, 1 en la cabeza y 1 en el cuello.
- **4 interruptores de pie**. 1 en cada pierna.
- **1 puerto IR**. Para la comunicación con otros Pleo y futuras formas de vida.
- **1 interruptor IR en la boca**. Para detectar objetos y la hoja.
- **1 sensor de orientación**. Para saber la posición de Pleo.
- **1 sensor de detección de batido**. Para detectar cuando se bate a Pleo.
- **2 micrófonos**.
- **1 cámara**.
- **1 puerto USB** y **1 batería**.

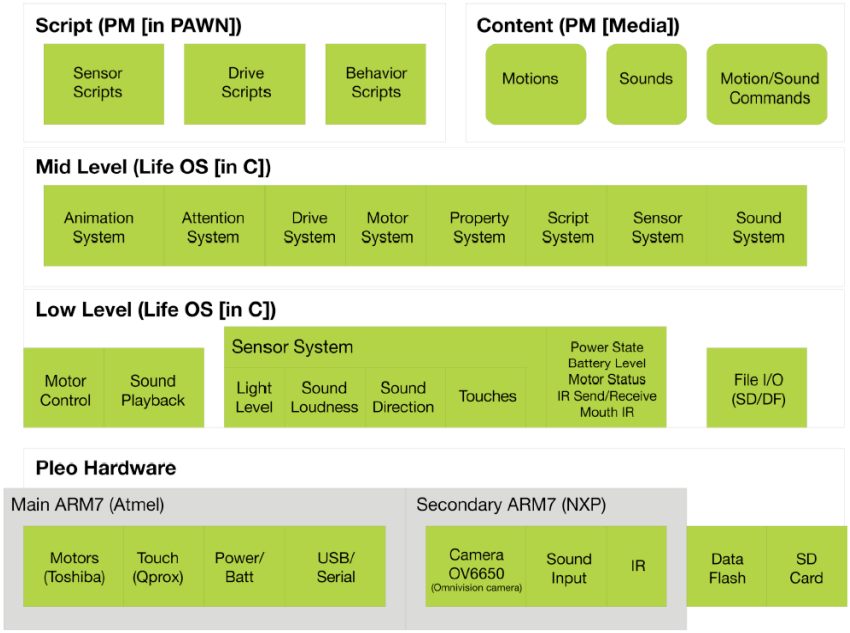


Figura 3.5 Componentes PLEO

En la Figura anterior se muestra las partes que constituyen el sistema operativo de Pleo, el Life OS, que lleva implementando en su firmware.

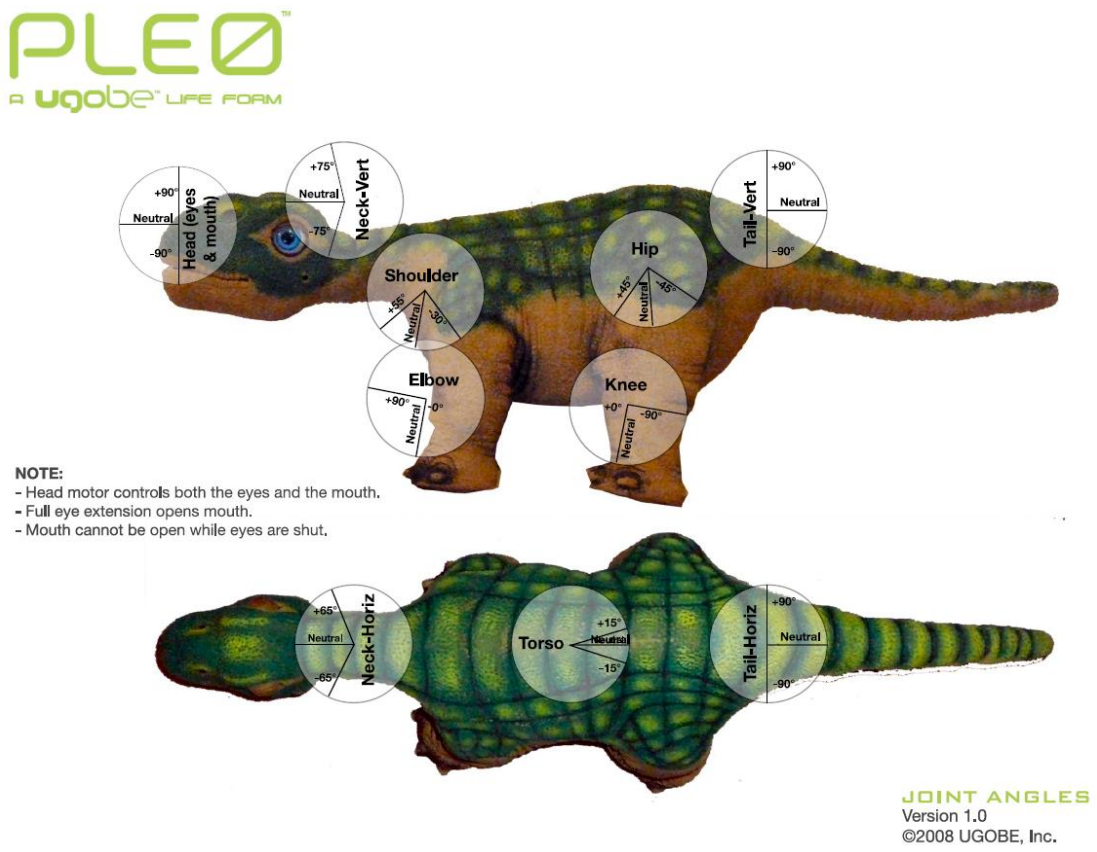


Figura 3.6 Pleo y los ángulos de movimiento de las articulaciones

En la anterior imagen se puede ver el aspecto de Pleo y los posibles movimientos con los ángulos de estos. Se trata del documento aportado por Ugobe.

En el presente proyecto Pleo es la forma de vida artificial para la que se creará inicialmente la herramienta utilizando su ranura SD. Se ha elegido este robot por diversas razones, principalmente por haber tenido bastante éxito y por lo tanto estar muy extendido. La documentación aportada por Ugobe también es un punto a favor pues todo está perfectamente documentado facilitando y animando a la creación de aplicaciones por parte de los usuarios. Por esta razón existe una gran comunidad de personas que comparten su afición a Pleo y sus conocimientos programándolo. El aspecto de Pleo también juega a su favor pues es más agradable que otras posibles opciones como AIBO.

Cabe destacar que el magazine Slate, hizo una prueba entre varios robots de este tipo y el único que convenció a los editores fue Pleo resultando el más natural y convincente de los evaluados

3.3.2 PDK

PDK, o Pleo Developers Kit (kit para desarrolladores de Pleo), es un conjunto de herramientas liberadas por Ugobe para facilitar a los desarrolladores el acceso a la API de Pleo incentivando así el desarrollo de aplicaciones de terceros para la programación de Pleo.

Este kit está compuesto por:

- Compilador de PAWN.
- Ficheros de recursos (sonidos, animaciones, etc.)
- Documentación para programadores.
- Ejemplos de aplicaciones para Pleo.
- Modelos en 3D de Pleo.

Un proyecto para Pleo consistirá en la compilación de varios ficheros resultando uno solo que se guardará en la tarjeta SD de Pleo para su posterior ejecución en el siguiente reinicio.

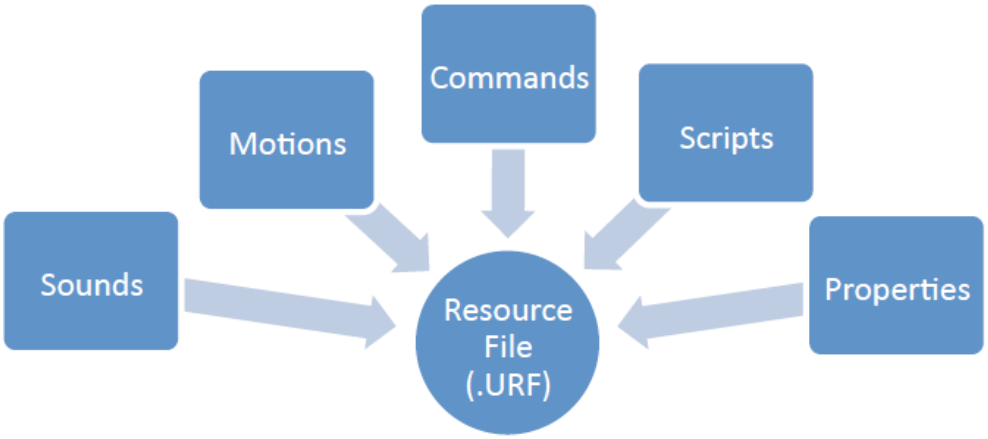


Figura 3.7 Recursos de un proyecto

Todos los recursos se definen en un archivo UPF como el siguiente:

```
<Ugobe_project name="Prueba">
	<options>
		<set name="top" value="../.." />
		<include value="./include:${top}/include" />
		<tools>
		    <pawn value="pawncc %i -V2048 -O2 -S64 -v2 -C- %I TARGET=100 -o%o" />
		</tools>
		<directories>
			<build value="build" />
			<include value="include" />
		</directories>
		<sound adpcm="true" />
		<motion version="3" />
		<folders />
	</options>
	<set-default name="MEDIA" value="media" />
	<set name="SOUNDS" value="${MEDIA}/sounds" />
	<set name="MOTIONS" value="${MEDIA}/motions" />
```

```
        <set name="SCRIPTS" value="${MEDIA}/scripts" />
        <resources>
                <!-- Sounds -->
                <!-- Motions -->
                <!-- Scripts -->
                <script path="sensors.p" />
        </resources>
</Ugobe_project>
```

En este archivo además se establecen las rutas por defecto donde el compilador ha de buscar los recursos e incluyo una línea con los parámetros a pasarle al compilador a la hora de hacer la llamada. En estos parámetros se incluyen opciones como la versión de la forma de vida para la que se va a compilar.

Además de este archivo UPF es necesario, como mínimo, un archivo con las órdenes. En el anterior ejemplo se puede ver como se ha declarado un script y el nombre de **sensors.p**, este archivo debe estar escrito en PAWN para que el compilador lo entienda, un ejemplo de dicho archivo se encuentra en la parte de PAWN de este mismo capítulo.

Estos archivos se pasan al compilador del PDK que devolverá un archivo ejecutable por Pleo, un archivo URF.

El PDK es una de las partes más importantes del proyecto, porque se usará su compilador para procesar los ficheros que deben ser ejecutados en Pleo. Como este Kit se distribuye conjuntamente con un gran número de recursos (sonidos, animaciones, etc.) se aprovecharan algunos de estos para añadirlos a la herramienta, ampliando el número de posibilidades.

3.4 Notaciones

3.4.1 UML

UML o Unified Modeling Language (lenguaje unificado de modelado) es un lenguaje gráfico de modelado estándar para describir la construcción y funcionamiento de un sistema. En 1997 se presento la versión 1.1 y se adoptó por la OMG (Object Management Group). 2005 fue el año en el que ISO aprobó UML como estándar en su versión 1.4.2.

UML hace uso de diferentes tipos de diagramas. Los diagramas pueden ser de estructura, comportamiento o interacción. Según lo que se vaya a representar y como se quiera mostrar se elige alguno de los diagramas de esas categorías. Los diagramas de estructuras se centran en mostrar los elementos que deben existir en el sistema, los de comportamiento que debe suceder y los de interacción que representa el flujo de control y datos.

La versión que utilizada para este proyecto es la 2.0 y será usada durante el desarrollo del proyecto. A lo largo de esta documentación se presentaran los diagramas realizados.

3.4.2 Métrica 3

Métrica es una metodología para la planificación, desarrollo y mantenimiento de los sistemas informáticos. Ofrece a las Organizaciones un instrumento útil para la sistematización de las actividades que dan soporte al ciclo de vida del software. Está promovida por el Ministerio de Administraciones Públicas de España. Está basada en el modelo de procesos del ciclo de vida de desarrollo ISO/IEC 12207.

Los objetivos de Métrica son proporcionar o definir Sistemas de Información que ayuden a conseguir los fines de la Organización mediante la definición de un marco estratégico para el desarrollo de los mismos, dotar a la Organización de productos software que satisfagan las necesidades de los usuarios dando una mayor importancia al análisis de requisitos, mejorar la productividad de los departamos de Sistemas y Tecnologías de la Información y las Comunicaciones, facilitar la comunicación y entendimiento entre los distintos participantes en la producción de software a lo largo del ciclo de vida del proyecto, facilitar la operación, mantenimiento y uso de los productos software obtenidos.

En este proyecto se hace uso de Métrica en su versión 3 de forma adaptada, conservando solo los apartados que se consideran más adecuados para este.

Capítulo 4. Planificación del Proyecto y Resumen de Presupuestos

4.1 Planificación

En esta planificación se muestran varias actividades y tareas a desarrollar para la buena ejecución del proyecto. Las tareas principales son **Investigación**, **Análisis**, **Diseño**, **Implementación**, **Pruebas** y **Documentación**.

La mayor parte de la **investigación** tiene lugar al principio del desarrollo de proyecto, para formarse una idea de lo que se tiene entre manos y la complejidad del proyecto. Para elaborar este proyecto se ha tenido que investigar sobre robótica, formas de vida artificiales, PLEO, alternativas en el mercado, C#, WPF y diseño de interfaces con WPF y XAML. En este proyecto la parte de la investigación tiene un gran peso debido a los conocimientos específicos necesarios para realizarlo.

El **análisis** tiene como objetivo delimitar qué se va a realizar y en este caso las tareas principales que se llevarán a cabo son la definición del sistema, la especificación de requisitos, diagrama de clases inicial, casos de uso y un diseño de interfaz previo.

Al finalizar la fase de análisis hay un hito para comprobar el desarrollo del proyecto y del progreso en cada una de las etapas.

El **diseño** aporta respuesta a cómo desarrollar el proyecto. En este caso, las principales tareas serán la de diseño de arquitectura de la herramienta y diseño de la interfaz. La interfaz tiene una gran importancia dado que la herramienta está destinada a todo tipo de personas.

Al finalizar el diseño hay otro hito para comprobar el progreso.

La **implementación** es la construcción de la herramienta y que tiene un gran peso temporal pues debe tener en cuenta todos los aspectos anteriores para su desarrollo.

Las **pruebas** es una de las últimas fases y consiste en la ejecución de diferentes pruebas para comprobar si satisface todo lo desarrollado en el análisis y diseño.

La **documentación** es un proceso que se realiza paralelamente a las anteriores tareas.

Por último el fin de proyecto es el último hito.

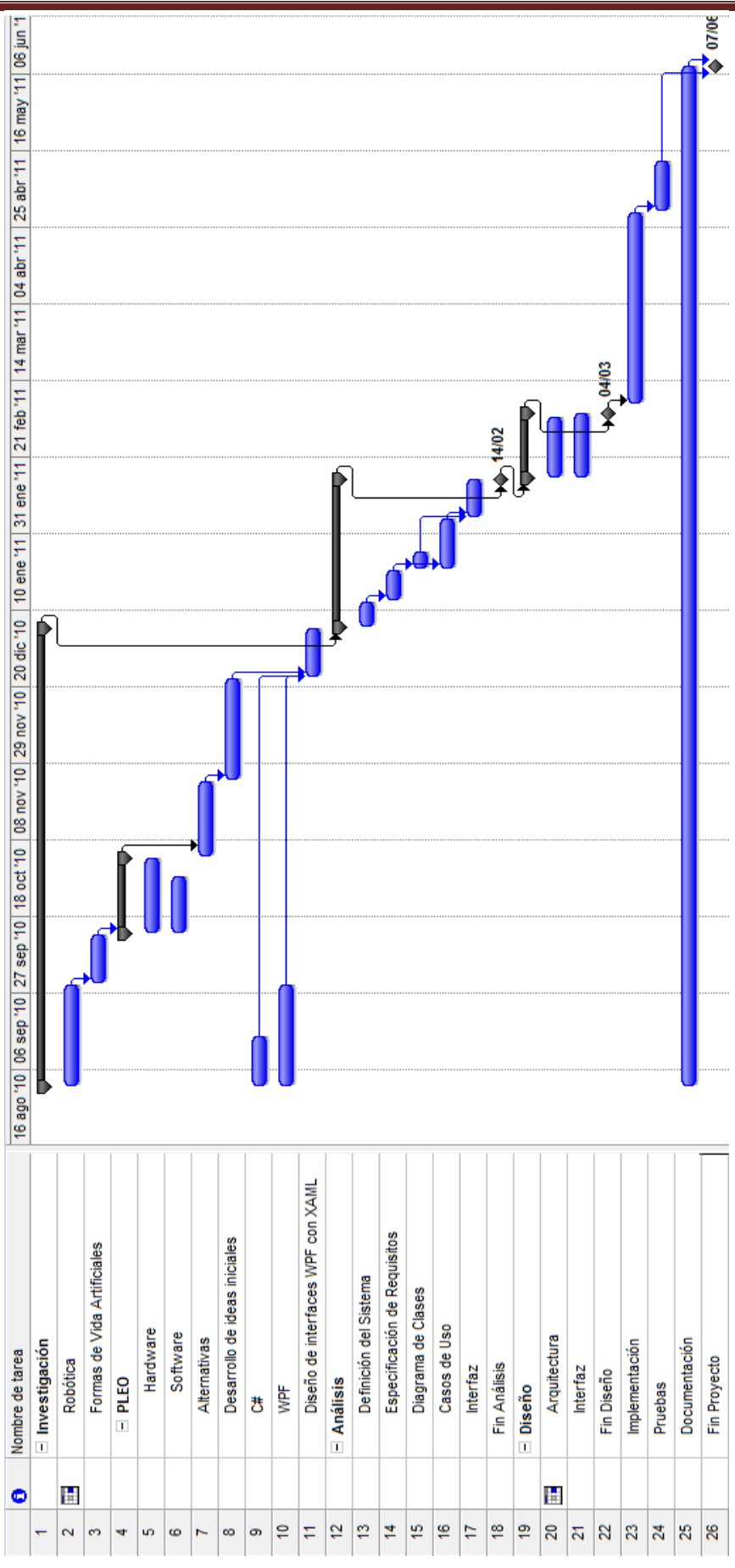


Figura 4.1 Diagrama de Gantt

4.2 Resumen del Presupuesto

A continuación se muestra un resumen del presupuesto.

Ítem	Concepto	Horas	Precio Unitario	Total
0	Investigación	180	30,00 €	5.400,00 €
1	Análisis	58	40,00 €	2.320,00 €
2	Diseño	28	40,00 €	1.120,00 €
3	Implementación	78	20,00 €	1.560,00 €
4	Pruebas	20	20,00 €	400,00 €
Ítem	**Concepto**	**Unidades**	**Precio Unitario**	**Total**
5	Pleo	2	300,00 €	600,00 €
6	Batería extra	1	50,00 €	50,00 €
			Subtotal	11.450,00 €
			IVA (18%)	2.061,00 €
			TOTAL	13.511,00 €

Figura 4.2 Resumen de presupuesto

Los gastos de cada uno de los conceptos se pueden ver gráficamente en el siguiente gráfico circular.

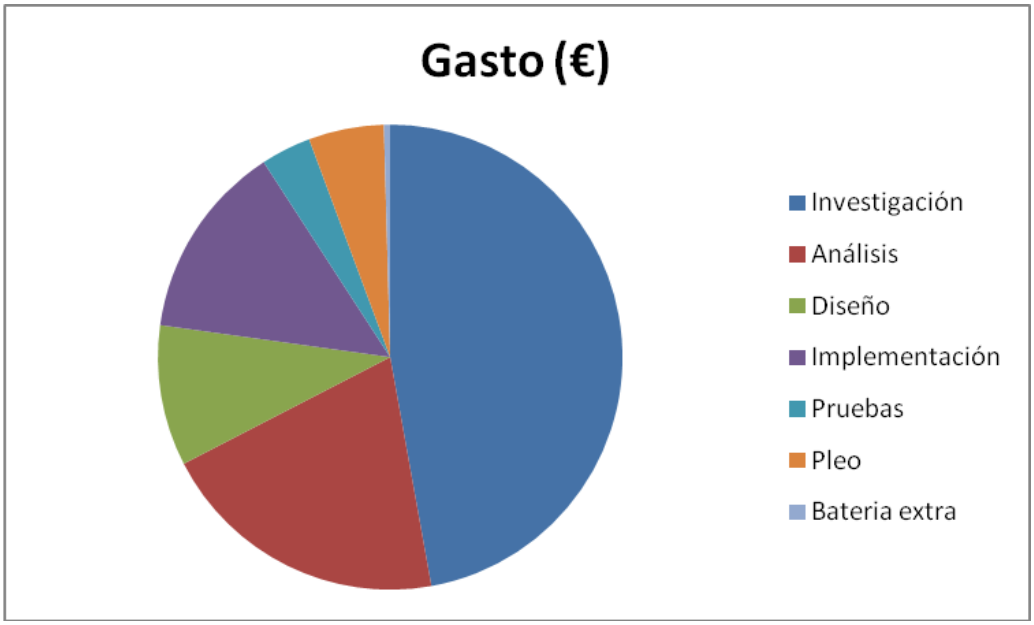


Figura 4.3 Gráfico circular con el gasto por ítems

Capítulo 5. Análisis

5.1 Definición del Sistema

5.1.1 Determinación del Alcance del Sistema

La herramienta de programación que se va a desarrollar pretende crear un entorno simple que facilite la programación de la forma de vida artificial a usuarios con escasos conocimientos tecnológicos. Esta herramienta estará compuesta por dos partes fundamentales: el módulo de programación y el compilador. El módulo de programación es la parte visual de la herramienta pues proporciona una interfaz y un área de trabajo en el que se crearán los programas, dentro de este módulo distinguimos otros dos sistemas, la interfaz y la personalidad. El compilador será el que traduzca el programa para la forma de vida pertinente.

La herramienta permitirá la creación desde el principio de una personalidad que posteriormente se podrá almacenar, descartar o instalar en la forma de vida artificial. En el caso de que se almacene se podrá abrir para su posterior edición en otro momento o para instalar dicho fichero en un nuevo robot.

El proceso de instalación conlleva ejecutar el compilado y la posterior acción de mover el archivo compilado a la ubicación deseada, idealmente una tarjeta SD que será la que se inserte en el robot.

Las posibles acciones disponibles que se podrán utilizar para la creación de los programas que ejecutará el robot están limitadas por las posibilidades que este ofrezca, en este caso Pleo. Las acciones disponibles son principalmente, entre otras, los movimientos de cada una de las extremidades con un ángulo determinado, la grabación y reproducción de sonidos, la toma de imágenes y la reproducción de animaciones. Además cada una de estas acciones podrá estar basada en decisiones o detección de eventos. Las decisiones que se tomen para un comportamiento u otro estarán basadas en elementos exteriores, tales como detección de obstáculos o huecos. También una acción se podrá programar en caso de que ocurra un evento concreto, como pudiera ser la detección de un toque en la piel del robot o la detección de un sonido. Además de estos condicionantes para ejecutar una acción existen otros, como el tiempo de inactividad.

El usuario tendrá la posibilidad de crear totalmente una personalidad a su gusto y también podrá elegir algunas de las acciones preestablecidas que se muestran en la herramienta.

5.2 Requisitos del Sistema

5.2.1 Obtención de los Requisitos del Sistema

5.2.1.1 Requisitos Funcionales

Código	Nombre Requisito	Descripción del Requisito
R1.1.1.1	Insertar Orden Animación	Se debe insertar la orden de movimiento eligiendo entre los predefinidos
R1.1.1.2	Editar Orden Animación	Se debe editar la orden de movimiento eligiendo entre los predefinidos
R1.1.1.3	Eliminar Orden Animación	Se debe eliminar la orden de movimiento predefinido
R1.1.2.1	Insertar Movimiento Ojos	Se debe insertar la orden de movimiento Ojos
R1.1.2.2	Editar Movimiento Ojos	Se debe editar la orden de movimiento Ojos
R1.1.2.3	Eliminar Movimiento Ojos	Se debe eliminar la orden de movimiento Ojos
R1.1.3.1.1	Insertar Movimiento Cadera Derecha	Se debe insertar la orden de movimiento Cadera Derecha
R1.1.3.1.2	Editar Movimiento Cadera Derecha	Se debe editar la orden de movimiento Cadera Derecha
R1.1.3.1.3	Eliminar Movimiento Cadera Derecha	Se debe eliminar la orden de movimiento Cadera Derecha
R1.1.3.2.1	Insertar Movimiento Cadera Izquierda	Se debe insertar la orden de movimiento Cadera Izquierda
R1.1.3.2.2	Editar Movimiento Cadera Izquierda	Se debe editar la orden de movimiento Cadera Izquierda
R1.1.3.2.3	Eliminar Movimiento Cadera Izquierda	Se debe eliminar la orden de movimiento Cadera Izquierda
R1.1.4.1.1	Insertar Movimiento Codo Derecho	Se debe insertar la orden de movimiento Codo Derecho
R1.1.4.1.2	Editar Movimiento Codo Derecho	Se debe editar la orden de movimiento Codo Derecho
R1.1.4.1.3	Eliminar Movimiento Codo Derecho	Se debe eliminar la orden de movimiento Codo Derecho
R1.1.4.2.1	Insertar Movimiento Codo Izquierdo	Se debe insertar la orden de movimiento Codo Izquierdo
R1.1.4.2.2	Editar Movimiento Codo Izquierdo	Se debe editar la orden de movimiento Codo Izquierdo
R1.1.4.2.3	Eliminar Movimiento Codo Izquierdo	Se debe eliminar la orden de movimiento Codo Izquierdo
R1.1.5.1	Insertar Movimiento Cola Horizontalmente	Se debe insertar la orden de movimiento Cola Horizontalmente
R1.1.5.2	Editar Movimiento Cola Horizontalmente	Se debe editar la orden de movimiento Cola Horizontalmente
R1.1.5.3	Eliminar Movimiento Cola Horizontalmente	Se debe eliminar la orden de movimiento Cola Horizontalmente

R1.1.6.1	Insertar Movimiento Cola Verticalmente	Se debe insertar la orden de movimiento Cola Verticalmente
R1.1.6.2	Editar Movimiento Cola Verticalmente	Se debe editar la orden de movimiento Cola Verticalmente
R1.1.6.3	Eliminar Movimiento Cola Verticalmente	Se debe eliminar la orden de movimiento Cola Verticalmente
R1.1.7.1	Insertar Movimiento Cuello Verticalmente	Se debe insertar la orden de movimiento Cuello Verticalmente
R1.1.7.2	Editar Movimiento Cuello Verticalmente	Se debe editar la orden de movimiento Cuello Verticalmente
R1.1.7.3	Eliminar Movimiento Cuello Verticalmente	Se debe eliminar la orden de movimiento Cuello Verticalmente
R1.1.8.1	Insertar Movimiento Cuello Horizontalmente	Se debe insertar la orden de movimiento Cuello Horizontalmente
R1.1.8.2	Editar Movimiento Cuello Horizontalmente	Se debe editar la orden de movimiento Cuello Horizontalmente
R1.1.8.3	Eliminar Movimiento Cuello Horizontalmente	Se debe eliminar la orden de movimiento Cuello Horizontalmente
R1.1.9.1.1	Insertar Movimiento Hombro Derecho	Se debe insertar la orden de movimiento Hombro Derecho
R1.1.9.1.2	Editar Movimiento Hombro Derecho	Se debe editar la orden de movimiento Hombro
R1.1.9.1.3	Eliminar Movimiento Hombro Derecho	Se debe eliminar la orden de movimiento Hombro Derecho
R1.1.9.2.1	Insertar Movimiento Hombro Izquierdo	Se debe insertar la orden de movimiento Hombro Izquierdo
R1.1.9.2.2	Editar Movimiento Hombro Izquierdo	Se debe editar la orden de movimiento Hombro Izquierdo
R1.1.9.2.3	Eliminar Movimiento Hombro Izquierdo	Se debe eliminar la orden de movimiento Hombro Izquierdo
R1.1.10.1.1	Insertar Movimiento Rodilla Derecha	Se debe insertar la orden de movimiento Rodilla Derecha
R1.1.10.1.2	Editar Movimiento Rodilla Derecha	Se debe editar la orden de movimiento Rodilla Derecha
R1.1.10.1.3	Eliminar Movimiento Rodilla Derecha	Se debe eliminar la orden de movimiento Rodilla Derecha
R1.1.10.2.1	Insertar Movimiento Rodilla Izquierda	Se debe insertar la orden de movimiento Rodilla Izquierda
R1.1.10.2.2	Editar Movimiento Rodilla Izquierda	Se debe editar la orden de movimiento Rodilla Izquierda
R1.1.10.2.3	Eliminar Movimiento Rodilla Izquierda	Se debe eliminar la orden de movimiento Rodilla Izquierda
R1.1.11.1	Insertar Movimiento Torso	Se debe insertar la orden de movimiento Torso
R1.1.11.2	Editar Movimiento Torso	Se debe editar la orden de movimiento Torso
R1.1.11.3	Eliminar Movimiento Torso	Se debe eliminar la orden de movimiento Torso
R1.2.1.1.1	Insertar Evento Táctil Cabeza	Se debe insertar el evento táctil Cabeza
R1.2.1.1.2	Eliminar Evento Táctil Cabeza	Se debe eliminar el evento táctil Cabeza

R1.2.1.2.1	Insertar Evento Táctil Mentón	Se debe insertar el evento táctil Mentón
R1.2.1.2.2	Eliminar Evento Táctil Mentón	Se debe eliminar el evento táctil Mentón
R1.2.1.3.1	Insertar Evento Táctil Espalda	Se debe insertar el evento táctil Espalda
R1.2.1.3.2	Eliminar Evento Táctil Espalda	Se debe eliminar el evento táctil Espalda
R1.2.1.4.1	Insertar Evento Táctil Cuello	Se debe insertar el evento táctil Cuello
R1.2.1.4.2	Eliminar Evento Táctil Cuello	Se debe eliminar el evento táctil Cuello
R1.2.1.5.1.1	Insertar Evento Táctil Pata Delantera Derecha	Se debe insertar el evento táctil Pata Delantera Derecha
R1.2.1.5.1.2	Eliminar Evento Táctil Pata Delantera Derecha	Se debe eliminar el evento táctil Pata Delantera Derecha
R1.2.1.5.2.1	Insertar Evento Táctil Pata Delantera Izquierda	Se debe insertar el evento táctil Pata Delantera Izquierda
R1.2.1.5.2.2	Eliminar Evento Táctil Pata Delantera Izquierda	Se debe eliminar el evento táctil Pata Delantera Izquierda
R1.2.1.5.3.1	Insertar Evento Táctil Pata Trasera Derecha	Se debe insertar el evento táctil Pata Trasera Derecha
R1.2.1.5.3.2	Eliminar Evento Táctil Pata Trasera Derecha	Se debe eliminar el evento táctil Pata Trasera Derecha
R1.2.1.5.4.1	Insertar Evento Táctil Pata Trasera Izquierda	Se debe insertar el evento táctil Pata Trasera Izquierda
R1.2.1.5.4.2	Eliminar Evento Táctil Pata Trasera Izquierda	Se debe eliminar el evento táctil Pata Trasera Izquierda
R1.2.2.1.1	Insertar Evento Interruptor Pata Delantera Derecha	Se debe insertar el evento de pulsación de interruptor Pata Delantera Derecha
R1.2.2.1.2	Eliminar Evento Interruptor Pata Delantera Derecha	Se debe eliminar el evento de pulsación de interruptor Pata Delantera Derecha
R1.2.2.2.1	Insertar Evento Interruptor Pata Delantera Izquierda	Se debe insertar el evento de pulsación de interruptor Pata Delantera Izquierda
R1.2.2.2.2	Eliminar Evento Interruptor Pata Delantera Izquierda	Se debe eliminar el evento de pulsación de interruptor Pata Delantera Izquierda
R1.2.2.3.1	Insertar Evento Interruptor Pata Trasera Derecha	Se debe insertar el evento de pulsación de interruptor Pata Trasera Derecha
R1.2.2.3.2	Eliminar Evento Interruptor Pata Trasera Derecha	Se debe eliminar el evento de pulsación de interruptor Pata Trasera Derecha
R1.2.2.4.1	Insertar Evento Interruptor Pata Trasera Izquierda	Se debe insertar el evento de pulsación de interruptor Pata Trasera Izquierda
R1.2.2.4.2	Eliminar Evento Interruptor Pata Trasera Izquierda	Se debe eliminar el evento de pulsación de interruptor Pata Trasera Izquierda
R1.2.3.1	Insertar Evento Obstáculo	Se debe insertar el evento de detección de obstáculo
R1.2.3.2	Eliminar Evento Obstáculo	Se debe eliminar el evento de detección de obstáculo
R1.2.4.1	Insertar Evento Filo	Se debe insertar el evento de detección de filo
R1.2.4.2	Eliminar Evento Filo	Se debe editar el evento de detección de filo

R1.2.5.1	Insertar Evento Sonido	Se debe insertar el evento de detección de sonido
R1.2.5.2	Eliminar Evento Sonido	Se debe eliminar el evento de detección de sonido
R1.2.6.1	Insertar Evento Posado	Se debe insertar el evento detección posado
R1.2.6.2	Eliminar Evento Posado	Se debe eliminar el evento de detección posado
R1.2.7.1	Insertar Evento Agitado	Se debe insertar el evento de detección de agitado
R1.2.7.2	Eliminar Evento Agitado	Se debe eliminar el evento de detección de agitado
R1.2.8.1	Insertar Evento Abuso	Se debe insertar el evento de detección de abuso
R1.2.8.2	Eliminar Evento Abuso	Se debe eliminar el evento de detección de abuso
R1.2.9.1	Insertar Evento Alzado	Se debe insertar el evento de detección de alzado
R1.2.9.2	Eliminar Evento Alzado	Se debe eliminar el evento de detección de alzado
R1.2.10.1	Insertar Evento Objeto Boca	Se debe insertar el evento de detección de objeto en la boca
R1.2.10.2	Eliminar Evento Objeto Boca	Se debe eliminar el evento de detección de objeto en la boca
R1.2.11.1	Insertar Evento Variación Luz	Se debe insertar el evento de detección de cambio significativo de niveles de luz
R1.2.11.2	Eliminar Evento Variación Luz	Se debe eliminar el evento de detección de cambio significativo de niveles de luz
R1.2.12.1	Insertar Evento Botón Wake Up	Se debe insertar el evento de pulsación del botón wake up
R1.2.12.2	Eliminar Evento Botón Wake Up	Se debe eliminar el evento de pulsación del botón wake up
R1.2.13.1	Insertar Evento Detección de Pleo	Se debe insertar el evento de detección de otro Pleo
R1.2.13.2	Eliminar Evento Detección de Pleo	Se debe eliminar el evento de detección de otro Pleo
R1.2.14.1	Insertar Evento Encendido	Se debe insertar el evento de Encendido
R1.2.14.2	Eliminar Evento Encendido	Se debe eliminar el evento Encendido
R1.3.1	Insertar Posición Neutra	Se debe insertar la orden de posición neutra
R1.3.2	Eliminar Posición Neutra	Se debe eliminar la orden de posición neutra
R1.4.1	Insertar Captura Imagen	Se debe insertar la orden de captura de una imagen
R1.4.2	Editar Captura Imagen	Se debe editar la orden de captura de una imagen
R1.4.3	Eliminar Captura Imagen	Se debe eliminar la orden de captura de una imagen
R1.5.1	Insertar Captura Sonido	Se debe insertar la orden de captura de sonido
R1.5.2	Editar Captura Sonido	Se debe editar la orden de captura de sonido

R1.5.3	Eliminar Captura Sonido	Se debe eliminar la orden de captura de sonido
R1.6.1	Insertar Reproducción Sonido	Se debe insertar la orden de reproducción de sonido
R1.6.2	Editar Reproducción Sonido	Se debe editar la orden de reproducción de sonido
R1.6.3	Eliminar Reproducción Sonido	Se debe eliminar la orden de reproducción de sonido
R1.7.1	Insertar Tiempo Espera	Se debe insertar la orden de tiempo en espera
R1.7.2	Editar Tiempo Espera	Se debe editar la orden de tiempo en espera
R1.7.3	Eliminar Tiempo Espera	Se debe eliminar la orden de tiempo en espera
R2.1	Aumentar turno	Se debe aumentar el turno de la acción seleccionada
R2.2	Disminuir turno	Se debe disminuir el turno de la acción seleccionada
R3.1	Nueva Personalidad	Se debe abrir un nuevo proyecto vacío
R3.2	Guardar Personalidad	Se debe salvar la personalidad actual
R3.3	Cargar Personalidad	Se debe cargar una personalidad previamente guardada
R4.1	Compilar	Se debe compilar para que la forma de vida pueda ejecutar el programa
R4.2	Indicar Forma de Vida	Se debe indicar para que forma de vida se va a compilar el programa
R4.3	Indicar Versión	Se debe indicar la versión de la forma de vida
R4.4	Indicar Ruta	Se debe indicar la ruta donde se guardará el ejecutable
R5.1	Editar Preferencias	Se deben editar y almacenar las preferencias del usuario
R5.2	Salvar LOG	Se debe editar y almacenar si se desea guardar el log del compilado
R5.3	Ruta LOG	Se debe editar y almacenar la ruta donde se guardará el archivo de LOG
R6	Mostrar Ayuda	Proporciona Ayuda al usuario cuando la solicite

5.2.1.2 Requisitos No Funcionales

5.2.1.2.1 Requisitos de Interfaz

La interfaz debe ser muy sencilla e intuitiva para cualquier tipo de usuario, experto o no, que la utilice. No deberá presentar ninguna ambigüedad en sus controles y su presentación deberá de verse de forma similar en diferentes sistemas.

La interfaz deberá evitar errores en la programación para que no se propaguen dando coherencia a todo el proceso.

Estará diseñada siguiendo las directrices generales para diseño de interfaces referidas a la accesibilidad, buena estética, etc.

5.2.1.2.2 Requisitos de Usuario

No es necesario que el usuario que utilice la herramienta posea conocimientos especiales, un niño con escasos conocimientos informáticos podrá trabajar con la herramienta. Se proporcionará una pequeña guía o ayuda para orientar los pasos básicos en la herramienta.

Lo único que se debe conocer son las funciones disponibles de la forma de vida a programar y sus características para adecuar el diagrama que se realice.

5.2.1.2.3 Requisitos Tecnológicos

Para ejecutar la herramienta es necesario un computador con los siguientes requisitos:

- Sistema Operativo Windows con .NET Framework instalado (Versión 4)
- Espacio suficiente en el disco duro para guardar los archivos
- Entrada para tarjetas SD o un lector USB de tarjetas.
- Kit de desarrollo de la forma de vida artificial que se use.

Además para ejecutar la aplicación que se desarrolle será necesaria una forma de vida artificial, en este caso Pleo, a la que se pueda insertar la tarjeta con el programa desarrollado.

5.2.2 Identificación de Actores del Sistema

Inicialmente solo se contempla un tipo de usuario, que será el que utilice la herramienta.

5.2.2.1 Actor 1: Usuario

Es el usuario que trabaja con la herramienta creando los diagramas que posteriormente se compilaran e instalarán en la forma de vida artificial.

Tiene la posibilidad de insertar, editar o eliminar todo tipo de acciones sin ninguna restricción.

5.2.3 Especificación de Casos de Uso

A continuación se muestran los diagramas de casos de uso, por su elevado número se mostrarán en diagramas separados y ordenados por categorías.

La siguiente figura muestra el modelo de casos de uso para insertar la orden de movimientos predefinidos.

A continuación se muestran los diagramas de casos de uso de los movimientos de articulaciones que se han separado debido al número de órdenes existentes.

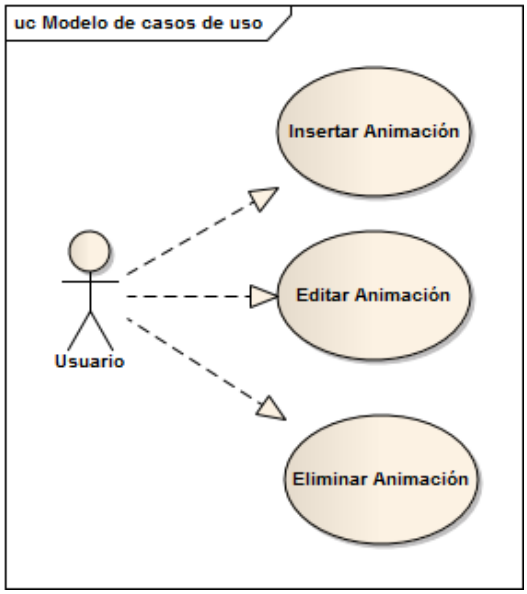


Figura 5.1 Casos de uso de animaciones

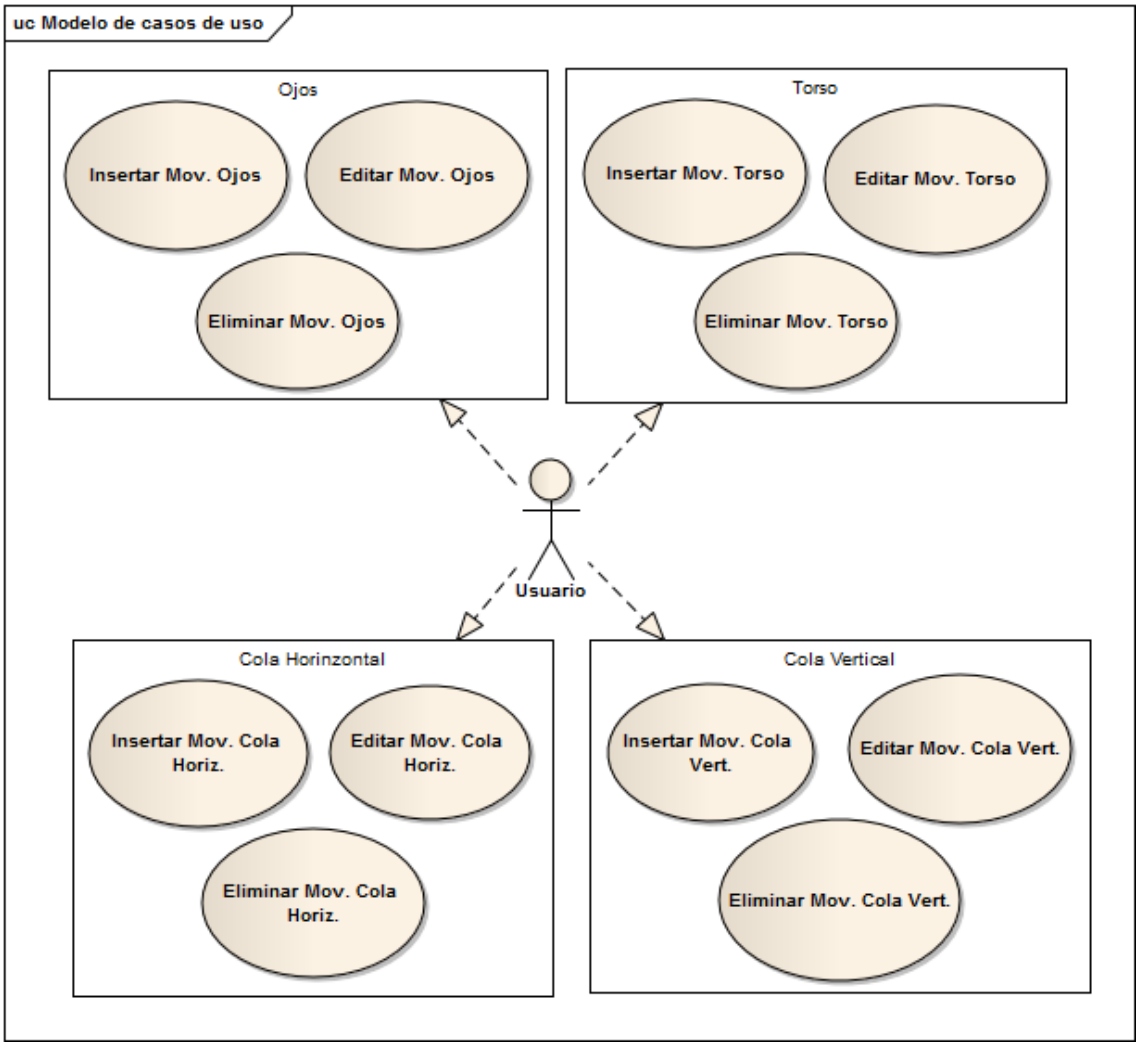


Figura 5.2 Casos de uso movimientos articulaciones I

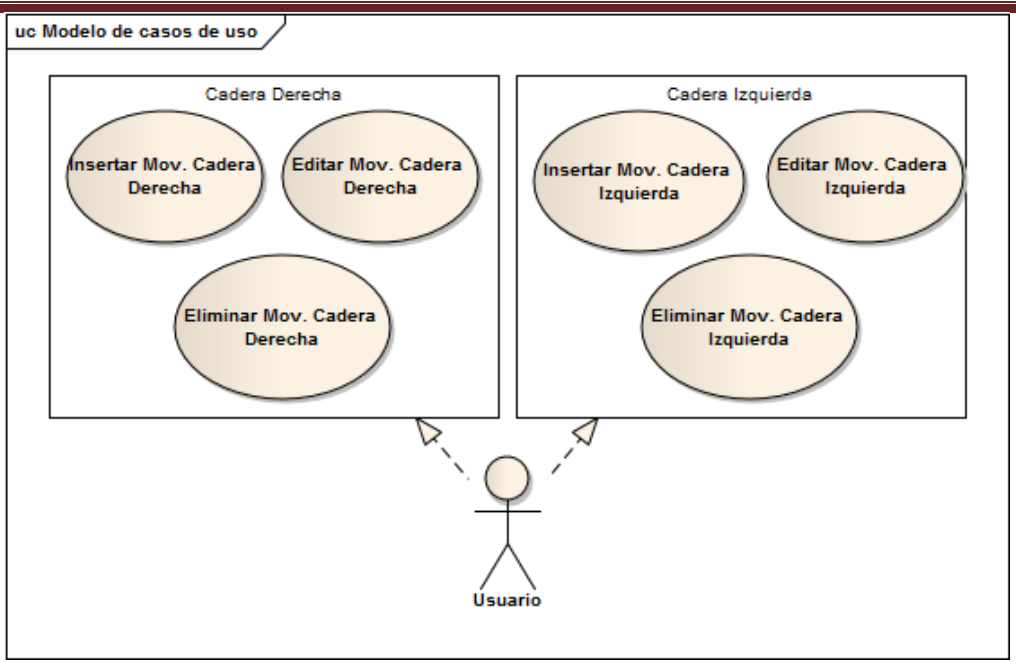


Figura 5.3 Casos de uso de movimientos articulaciones II

A continuación los movimientos de cuello (vert. y horiz.) y hombro (derecho e izquierdo). El hombro se corresponde con la articulación superior de las patas delanteras.

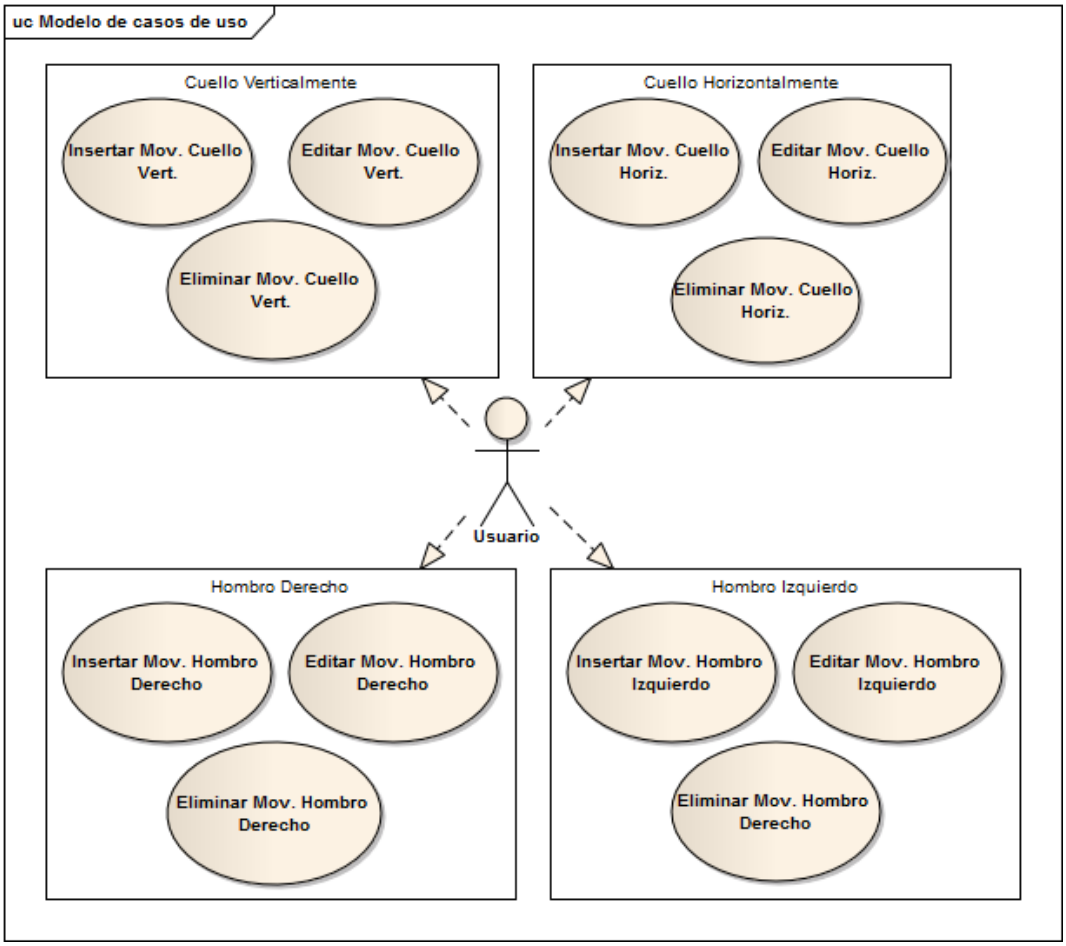


Figura 5.4 Casos de uso de movimientos articulaciones III

En el siguiente diagrama se representan las órdenes de movimiento de rodilla derecha, rodilla izquierda, codo derecho y codo izquierdo. La rodilla se corresponde con las articulaciones inferiores de las patas traseras. El codo se corresponde con las articulaciones inferiores de las patas delanteras.

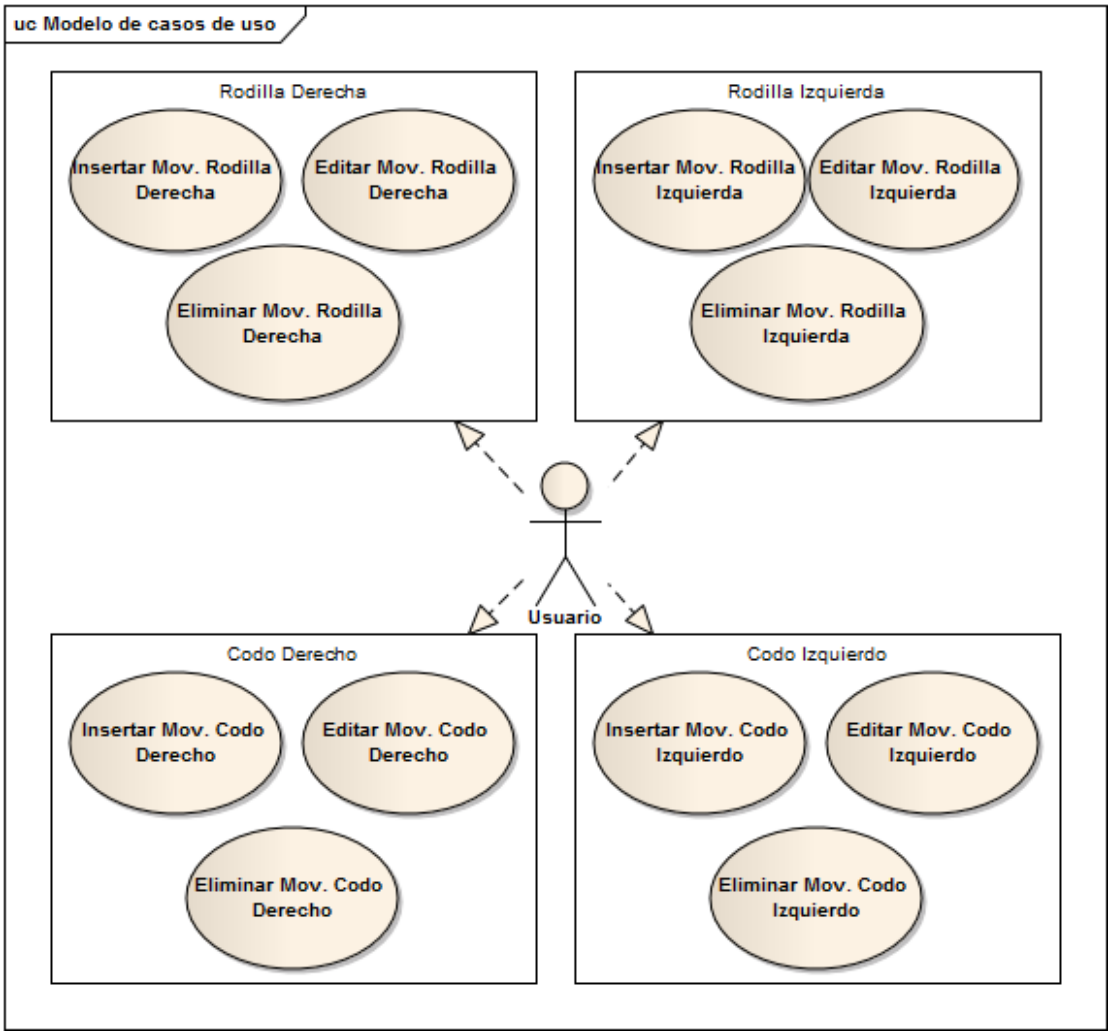


Figura 5.5 Casos de uso de movimientos articulaciones IV

En los siguientes diagramas se muestran los casos de uso de las operaciones con los eventos.

A continuación casos de uso de los eventos táctiles.

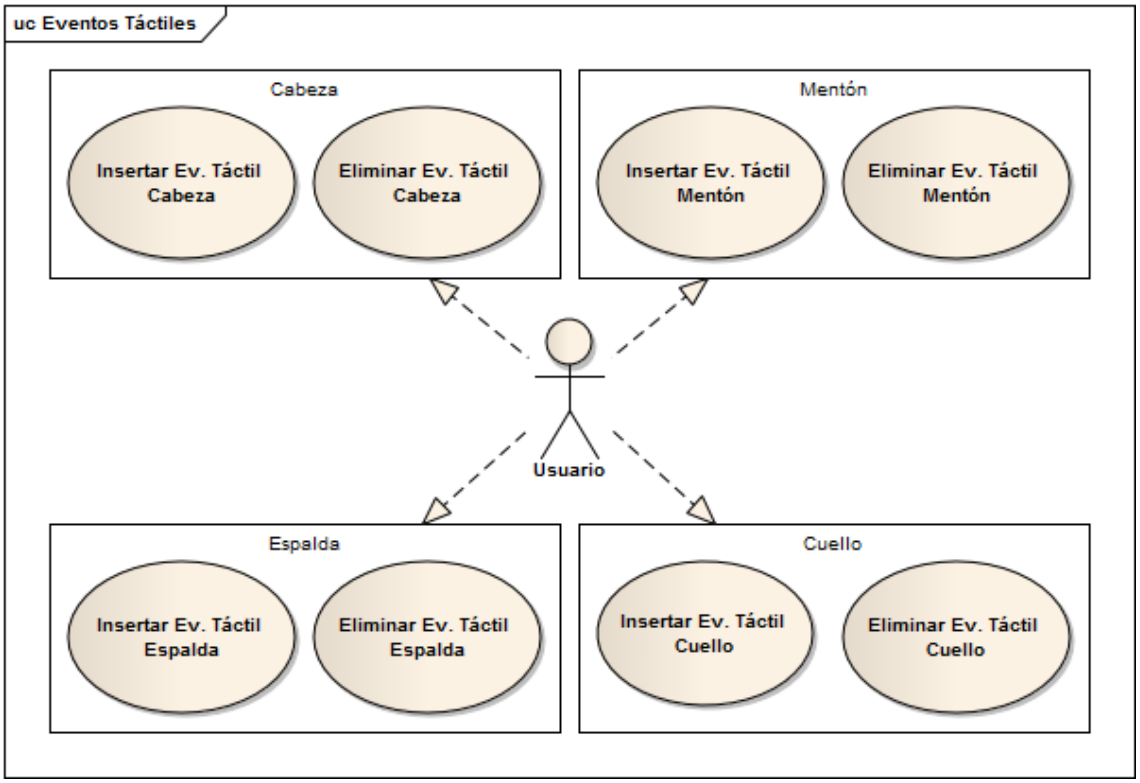


Figura 5.6 Casos de uso de eventos I

Debajo, diagrama de casos de uso de los eventos táctiles de las extremidades

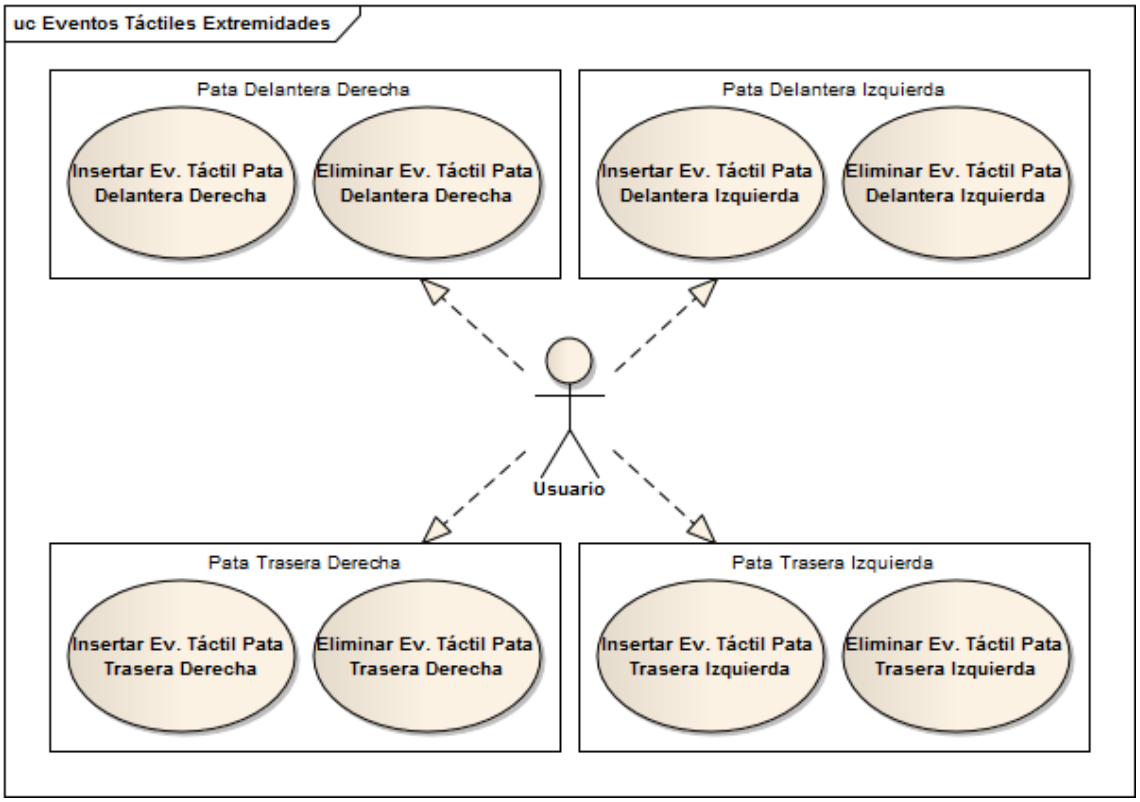


Figura 5.7 Casos de uso de eventos II

El siguiente diagrama muestra los casos de uso para los eventos de detección se obstáculos, filos y sonidos.

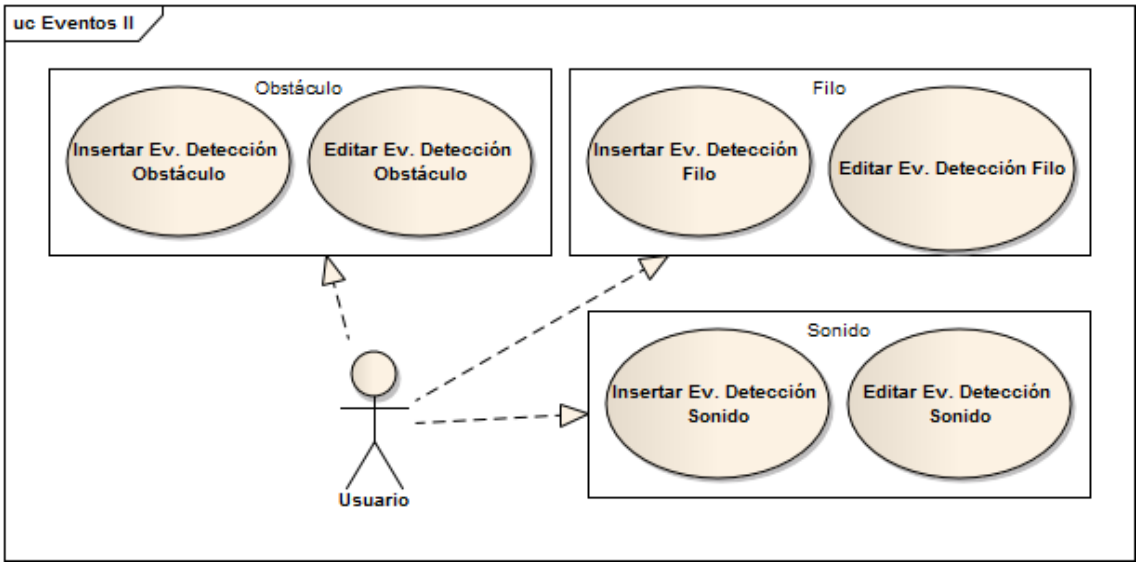


Figura 5.8 Casos de uso de eventos III

Cada pata posee un interruptor de pie y los eventos están representados en el siguiente diagrama.

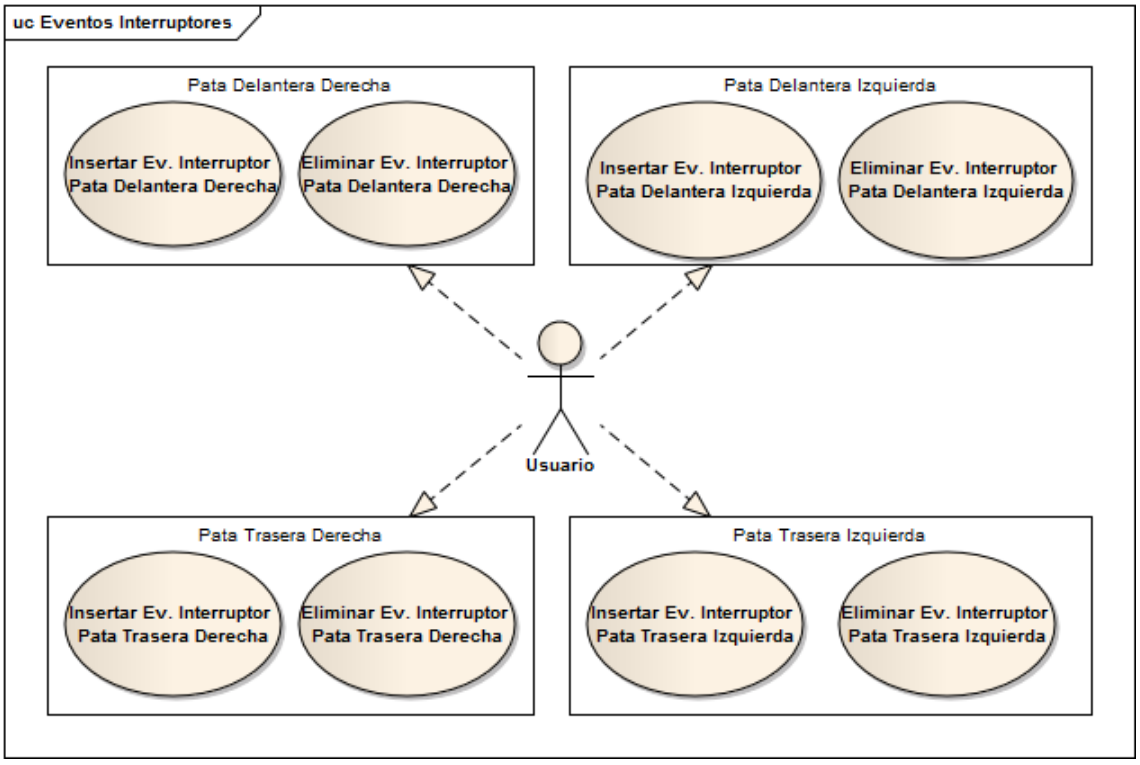


Figura 5.9 Casos de uso de eventos IV

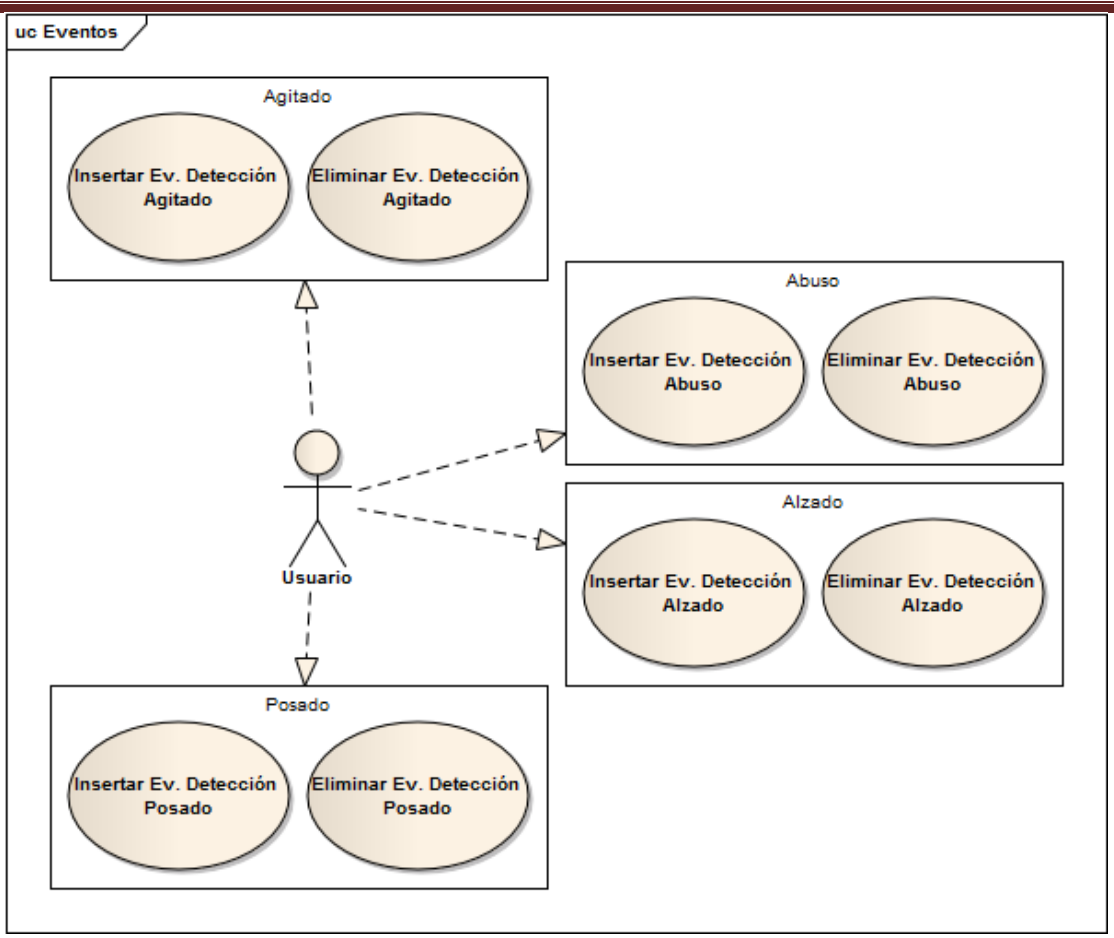


Figura 5.10 Casos de uso de eventos V

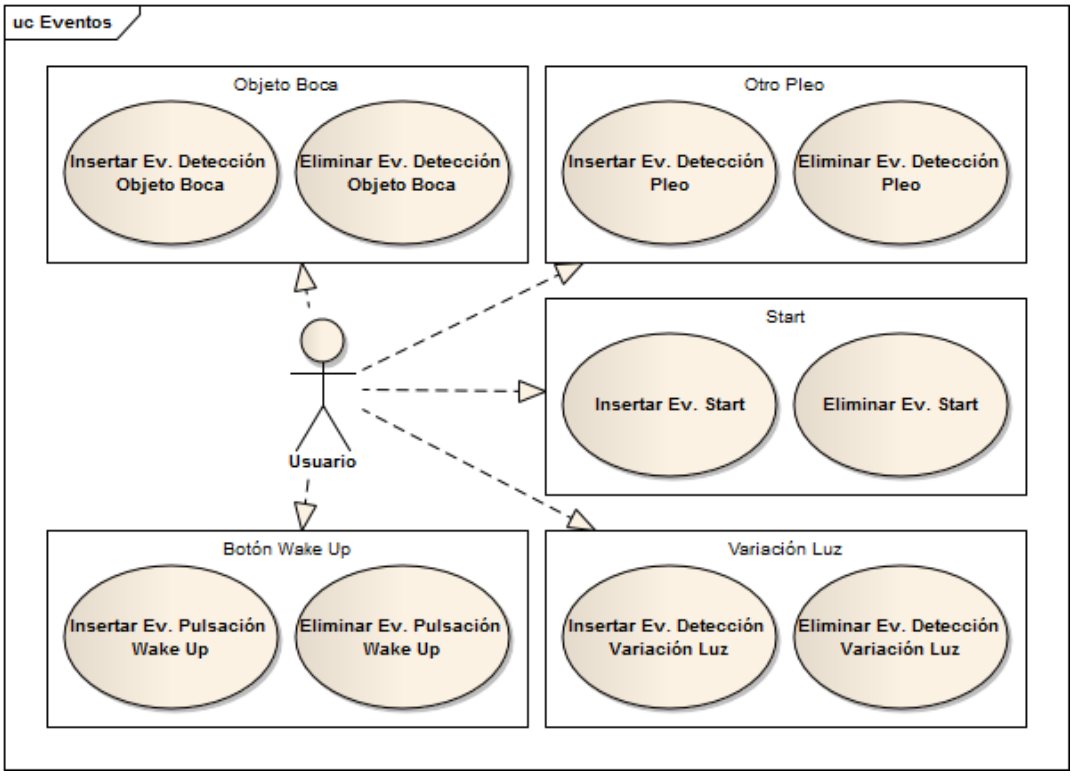


Figura 5.11 Casos de uso de las órdenes de eventos VI

Además de los movimientos hay otro tipo de órdenes posibles utilizando el hardware de Pleo, este diagrama representa dichas órdenes.

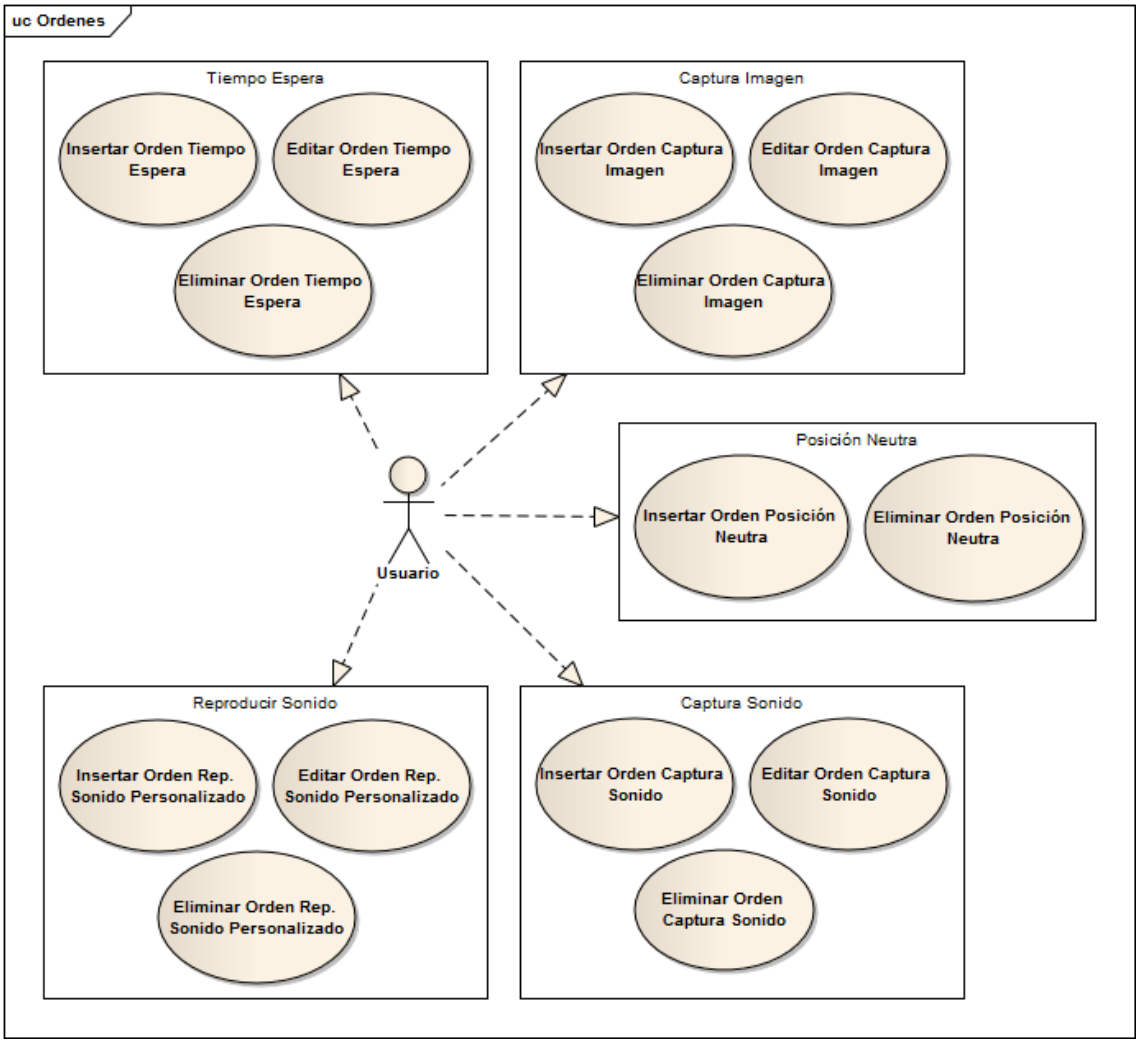


Figura 5.12 Casos de uso de las otras órdenes

Las opciones del programa permiten el siguiente diagrama con casos de uso.

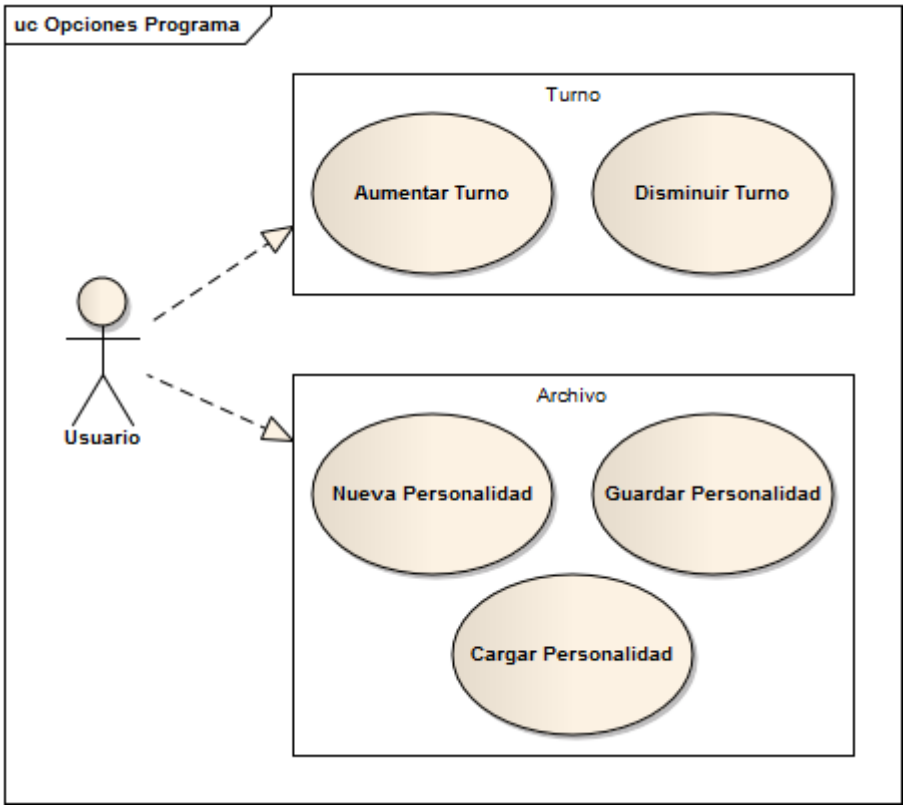


Figura 5.13 Casos de uso de órdenes del programa

El compilador tiene los siguientes casos de uso.

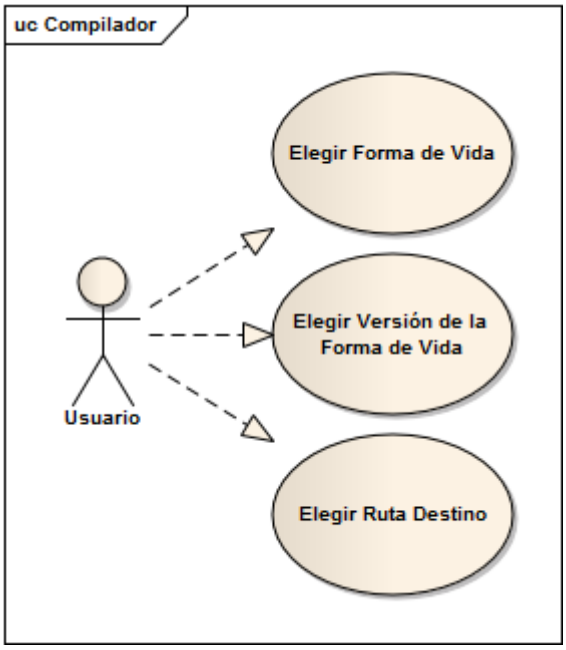


Figura 5.14 Caso de uso Compilador

A continuación se muestra el caso de uso para las preferencias, editarlas (guardarlas y recuperarlas) así como salvar el log y editar su ruta.

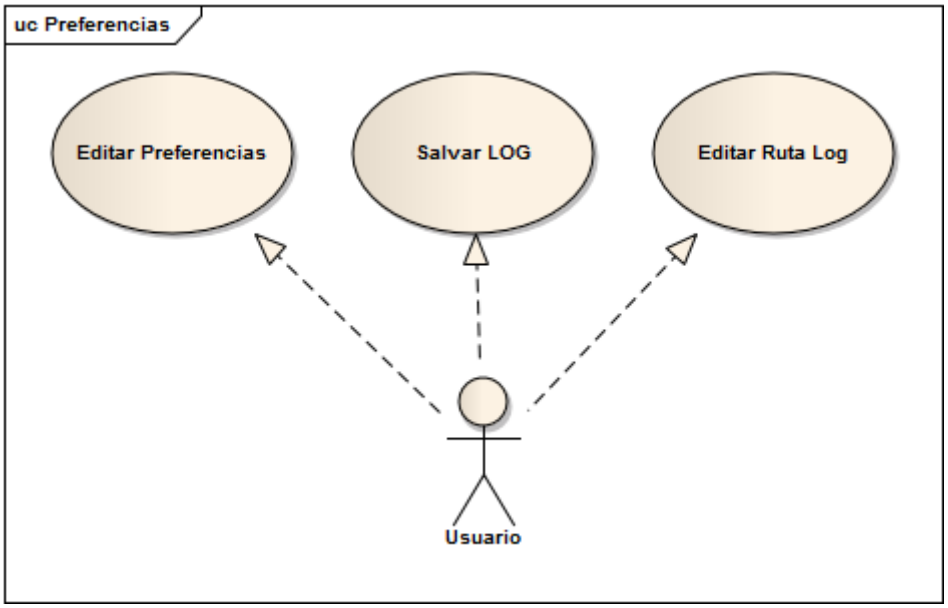


Figura 5.15 Caso de Uso de Preferencias

Por último el caso de uso de la ayuda.

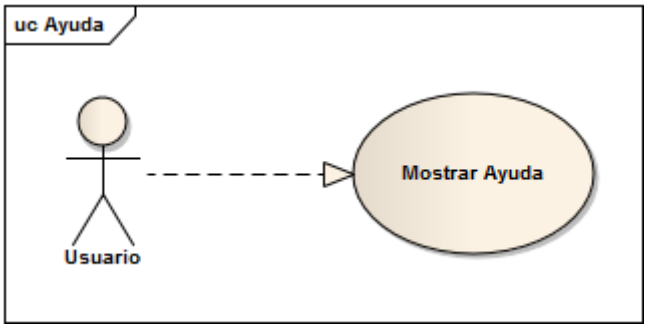


Figura 5.16 Caso de Uso Ayuda

Nombre del Caso de Uso
Insertar Orden Animación
Descripción
Inserción de una orden de animación. El sistema mostrará una ventana para elegir el archivo de animación en el equipo. Corresponde con el requisito R1.1.1.1

Nombre del Caso de Uso
Editar Orden Animación
Descripción
El usuario elegirá una orden ya insertada y la podrá editar para elegir otra animación. Corresponde con el requisito R1.1.1.2

Nombre del Caso de Uso
Eliminar Orden Animación
Descripción
El usuario elegirá la orden y la eliminará. Corresponde con el requisito R1.1.1.3

Nombre del Caso de Uso
Insertar Movimiento Ojos
Descripción
El usuario elige insertar la orden de movimiento de ojos y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.2.1

Nombre del Caso de Uso
Editar Movimiento Ojos
Descripción
El usuario elige una orden de movimiento de ojos para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.2.2

Nombre del Caso de Uso
Eliminar Movimiento Ojos
Descripción
El usuario elige una orden de movimiento de ojos y la borra. Corresponde con el requisito R1.1.2.3

Nombre del Caso de Uso
Insertar Movimiento Cadera Derecha
Descripción
El usuario elige insertar la orden de movimiento de cadera derecha y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.3.1.1

Nombre del Caso de Uso
Editar Movimiento Cadera Derecha
Descripción
El usuario elige una orden de movimiento de cadera derecha para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.3.1.1.2

Nombre del Caso de Uso
Eliminar Movimiento Cadera Derecha
Descripción
El usuario elige una orden de movimiento de cadera derecha y la borra. Corresponde con el requisito R1.1.3.1.3

Nombre del Caso de Uso
Insertar Movimiento Cadera Izquierda
Descripción
El usuario elige insertar la orden de movimiento de cadera izquierda y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.3.2.1

Nombre del Caso de Uso
Editar Movimiento Cadera Izquierda
Descripción
El usuario elige una orden de movimiento de cadera izquierda para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.3.2.2

Nombre del Caso de Uso
Eliminar Movimiento Cadera Izquierda
Descripción
El usuario elige una orden de movimiento de cadera izquierda y la borra. Corresponde con el requisito R1.1.3.2.3

Nombre del Caso de Uso
Insertar Movimiento Codo Derecho
Descripción
El usuario elige insertar la orden de movimiento de codo derecho y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.4.1.1

Nombre del Caso de Uso
Editar Movimiento Codo Derecho
Descripción
El usuario elige una orden de movimiento de codo derecho para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.4.1.2

Nombre del Caso de Uso
Eliminar Movimiento Codo Derecho
Descripción
El usuario elige una orden de movimiento de codo derecho y la borra. Corresponde con el requisito R1.1.4.1.3

Nombre del Caso de Uso
Insertar Movimiento Codo Izquierdo
Descripción
El usuario elige insertar la orden de movimiento de codo izquierdo y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.4.2.1

Nombre del Caso de Uso
Editar Movimiento Codo Izquierdo
Descripción
El usuario elige una orden de movimiento de codo izquierdo para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.4.2.2

Nombre del Caso de Uso
Eliminar Movimiento Codo Izquierdo
Descripción
El usuario elige una orden de movimiento de codo izquierdo y la borra. Corresponde con el requisito R1.1.4.2.3

Nombre del Caso de Uso
Insertar Movimiento Cola Horizontalmente
Descripción
El usuario elige insertar la orden de movimiento de cola horizontalmente y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.5.1

Nombre del Caso de Uso
Editar Movimiento Cola Horizontalmente
Descripción
El usuario elige una orden de movimiento de cola horizontalmente para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.5.2

Nombre del Caso de Uso
Eliminar Movimiento Cola Horizontalmente
Descripción
El usuario elige una orden de movimiento de cola horizontalmente y la borra. Corresponde con el requisito R1.1.5.3

Nombre del Caso de Uso
Insertar Movimiento Cola Verticalmente
Descripción
El usuario elige insertar la orden de movimiento de cola verticalmente y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.6.1

Nombre del Caso de Uso
Editar Movimiento Cola Verticalmente
Descripción
El usuario elige una orden de movimiento de cola verticalmente para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.6.2

Nombre del Caso de Uso
Eliminar Movimiento Cola Verticalmente
Descripción
El usuario elige una orden de movimiento de cola verticalmente y la borra. Corresponde con el requisito R1.1.6.3

Nombre del Caso de Uso
Insertar Movimiento Cuello Verticalmente
Descripción
El usuario elige insertar la orden de movimiento de cuello verticalmente y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.7.1

Nombre del Caso de Uso
Editar Movimiento Cuello Verticalmente
Descripción
El usuario elige una orden de movimiento de cuello verticalmente para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.7.2

Nombre del Caso de Uso
Eliminar Movimiento Cuello Verticalmente
Descripción
El usuario elige una orden de movimiento de cuello verticalmente y la borra. Corresponde con el requisito R1.1.7.3

Nombre del Caso de Uso
Insertar Movimiento Cuello Horizontalmente
Descripción
El usuario elige insertar la orden de movimiento de cuello horizontalmente y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.8.1

Nombre del Caso de Uso
Editar Movimiento Cuello Horizontalmente
Descripción
El usuario elige una orden de movimiento de cuello horizontalmente para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.8.2

Nombre del Caso de Uso
Eliminar Movimiento Cuello Horizontalmente
Descripción
El usuario elige una orden de movimiento de cuello horizontalmente y la borra. Corresponde con el requisito R1.1.8.3

Nombre del Caso de Uso
Insertar Movimiento Hombro Derecho
Descripción
El usuario elige insertar la orden de movimiento de hombro derecho y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.9.1.1

Nombre del Caso de Uso
Editar Movimiento Hombro Derecho
Descripción
El usuario elige una orden de movimiento de hombro derecho para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.9.1.2

Nombre del Caso de Uso
Eliminar Movimiento Hombro Derecho
Descripción
El usuario elige una orden de movimiento de hombro derecho y la borra. Corresponde con el requisito R1.1.9.1.3

Nombre del Caso de Uso
Insertar Movimiento Hombro Izquierdo
Descripción
El usuario elige insertar la orden de movimiento de hombro izquierdo y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.9.2.1

Nombre del Caso de Uso
Editar Movimiento Hombro Izquierdo
Descripción
El usuario elige una orden de movimiento de hombro izquierdo para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.9.2.2

Nombre del Caso de Uso
Eliminar Movimiento Hombro Izquierdo
Descripción
El usuario elige una orden de movimiento de hombro izquierdo y la borra. Corresponde con el requisito R1.1.9.2.3

Nombre del Caso de Uso
Insertar Movimiento Rodilla Derecha
Descripción
El usuario elige insertar la orden de movimiento de rodilla derecha y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.10.1.1

Nombre del Caso de Uso
Editar Movimiento Rodilla Derecha
Descripción
El usuario elige una orden de movimiento de rodilla derecha para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.10.1.2

Nombre del Caso de Uso
Eliminar Movimiento Rodilla Derecha
Descripción
El usuario elige una orden de movimiento de rodilla derecha y la borra. Corresponde con el requisito R1.1.10.1.3

Nombre del Caso de Uso
Insertar Movimiento Rodilla Izquierda
Descripción
El usuario elige insertar la orden de movimiento de rodilla izquierda y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.10.2.1

Nombre del Caso de Uso
Editar Movimiento Rodilla Izquierda
Descripción
El usuario elige una orden de movimiento de rodilla izquierda para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.10.2.2

Nombre del Caso de Uso
Eliminar Movimiento Rodilla Izquierda
Descripción
El usuario elige una orden de movimiento de rodilla izquierda y la borra. Corresponde con el requisito R1.1.10.2.3

Nombre del Caso de Uso
Insertar Movimiento Torso
Descripción
El usuario elige insertar la orden de movimiento de torso y el sistema le mostrará un diálogo donde introducir los ángulos de dicho movimiento. Elegirá si desea que dicho movimiento sea repetitivo. Corresponde con el requisito R1.1.11.1

Nombre del Caso de Uso
Editar Movimiento Torso
Descripción
El usuario elige una orden de movimiento de torso para editar y el sistema le muestra los valores editables, en este caso los ángulos de movimiento y si desea que sea repetitivo. Corresponde con el requisito R1.1.11.2

Nombre del Caso de Uso
Eliminar Movimiento Torso
Descripción
El usuario elige una orden de movimiento de torso y la borra. Corresponde con el requisito R1.1.11.3

Nombre del Caso de Uso
Insertar Evento Táctil Cabeza
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor táctil de la cabeza detecte presión. Corresponde con el requisito R1.2.1.1.1

Nombre del Caso de Uso
Eliminar Evento Táctil Cabeza
Descripción
El usuario borra el evento táctil de cabeza. Corresponde con el requisito R1.2.1.1.3

Nombre del Caso de Uso
Insertar Evento Táctil Mentón
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor táctil del mentón detecte presión. Corresponde con el requisito R1.2.1.2.1

Nombre del Caso de Uso
Eliminar Evento Táctil Mentón
Descripción
El usuario borra el evento táctil del mentón. Corresponde con el requisito R1.2.1.2.3

Nombre del Caso de Uso
Insertar Evento Táctil Espalda
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor táctil de la espalda detecte presión. Corresponde con el requisito R1.2.1.3.1

Nombre del Caso de Uso
Eliminar Evento Táctil Espalda
Descripción
El usuario borra el evento táctil de la espalda. Corresponde con el requisito R1.2.1.3.3

Nombre del Caso de Uso
Insertar Evento Táctil Cuello
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor táctil del cuello detecte presión. Corresponde con el requisito R1.2.1.4.1

Nombre del Caso de Uso
Eliminar Evento Táctil Cuello
Descripción
El usuario borra el evento táctil del cuello. Corresponde con el requisito R1.2.1.4.3

Nombre del Caso de Uso
Insertar Evento Táctil Pata Delantera Derecha
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor táctil de la pata delantera derecha detecte presión. Corresponde con el requisito R1.2.1.5.1.1

Nombre del Caso de Uso
Eliminar Evento Táctil Pata Delantera Derecha
Descripción
El usuario borra el evento táctil de la pata delantera derecha. Corresponde con el requisito R1.2.1.5.1.2

Nombre del Caso de Uso
Insertar Evento Táctil Pata Delantera Izquierda
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor táctil de la panta delantera izquierda detecte presión. Corresponde con el requisito R1.2.1.5.2.1

Nombre del Caso de Uso
Eliminar Evento Táctil Pata Delantera Izquierda
Descripción
El usuario borra el evento táctil de la pata delantera izquierda. Corresponde con el requisito R1.2.1.5.2.2

Nombre del Caso de Uso
Insertar Evento Táctil Pata Trasera Derecha
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor táctil de la pata trasera derecha detecte presión. Corresponde con el requisito R1.2.1.5.3.1

Nombre del Caso de Uso
Eliminar Evento Táctil Pata Trasera Derecha
Descripción
El usuario borra el evento táctil de la pata trasera derecha. Corresponde con el requisito R1.2.1.5.3.2

Nombre del Caso de Uso
Insertar Evento Táctil Pata Trasera Derecha
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor táctil de la pata trasera derecha detecte presión. Corresponde con el requisito R1.2.1.5.4.1

Nombre del Caso de Uso
Eliminar Evento Táctil Pata Trasera Izquierda
Descripción
El usuario borra el evento táctil de la pata trasera izquierda. Corresponde con el requisito R1.2.1.5.4.2

Nombre del Caso de Uso
Insertar Evento Interruptor Pata Delantera Derecha
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor interruptor de la pata delantera derecha se presione. Corresponde con el requisito R1.2.2.1.1

Nombre del Caso de Uso
Eliminar Evento Interruptor Pata Delantera Derecha
Descripción
El usuario borra el evento interruptor de la pata delantera derecha. Corresponde con el requisito R1.2.2.1.2

Nombre del Caso de Uso
Insertar Evento Interruptor Pata Delantera Izquierda
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor interruptor de la pata delantera izquierda se presione. Corresponde con el requisito R1.2.2.2.1

Nombre del Caso de Uso
Eliminar Evento Interruptor Pata Delantera Izquierda
Descripción
El usuario borra el evento interruptor de la pata delantera izquierda. Corresponde con el requisito R1.2.2.2.2

Nombre del Caso de Uso
Insertar Evento Interruptor Pata Trasera Derecha
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor interruptor la pata trasera derecha se presione. Corresponde con el requisito R1.2.2.3.1

Nombre del Caso de Uso
Eliminar Evento Interruptor Pata Trasera Derecha
Descripción
El usuario borra el evento interruptor de la pata trasera derecha. Corresponde con el requisito R1.2.2.3.2

Nombre del Caso de Uso
Insertar Evento Interruptor Pata Trasera Izquierda
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando el sensor interruptor de la pata trasera izquierda se presione. Corresponde con el requisito R1.2.2.4.1

Nombre del Caso de Uso
Eliminar Evento Interruptor Pata Trasera Izquierda
Descripción
El usuario borra el evento interruptor de la pata trasera izquierda. Corresponde con el requisito R1.2.2.4.2

Nombre del Caso de Uso
Insertar Evento Obstáculo
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se detecte un obstáculo. Corresponde con el requisito R1.2.3.1

Nombre del Caso de Uso
Eliminar Evento Obstáculo
Descripción
El usuario borra el evento detección de obstáculo. Corresponde con el requisito R1.2.3.3

Nombre del Caso de Uso
Insertar Evento Filo
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se detecte un filo, como por ejemplo el final de una mesa. Corresponde con el requisito R1.2.4.1

Nombre del Caso de Uso
Eliminar Evento Filo
Descripción
El usuario borra el evento detección de filo. Corresponde con el requisito R1.2.4.3

Nombre del Caso de Uso
Insertar Evento Sonido
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se detecte un sonido, como por ejemplo una palmada. Corresponde con el requisito R1.2.5.1

Nombre del Caso de Uso
Eliminar Evento Sonido
Descripción
El usuario borra el evento detección de sonido. Corresponde con el requisito R1.2.5.3

Nombre del Caso de Uso
Insertar Evento Posado
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se detecte que el robot está posado. Corresponde con el requisito R1.2.6.1

Nombre del Caso de Uso
Eliminar Evento Posado
Descripción
El usuario borra el evento de posado. Corresponde con el requisito R1.2.6.2

Nombre del Caso de Uso
Insertar Evento Agitado
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se detecte que el robot está siendo agitado. Corresponde con el requisito R1.2.7.1

Nombre del Caso de Uso
Eliminar Evento Agitado
Descripción
El usuario borra el evento de agitado. Corresponde con el requisito R1.2.7.2

Nombre del Caso de Uso
Insertar Evento Abuso
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se detecte que el robot está siendo tratado de mal forma, por ejemplo cogiéndolo por la cola. Corresponde con el requisito R1.2.8.1

Nombre del Caso de Uso
Eliminar Evento Abuso
Descripción
El usuario borra el evento de abuso. Corresponde con el requisito R1.2.8.2

Nombre del Caso de Uso
Insertar Evento Alzado
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se detecte que el robot está alzado, por ejemplo cuando se coge en brazos. Corresponde con el requisito R1.2.9.1

Nombre del Caso de Uso
Eliminar Evento Alzado
Descripción
El usuario borra el evento de alzado. Corresponde con el requisito R1.2.9.2

Nombre del Caso de Uso
Insertar Evento Objeto Boca
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se detecte en el interior de la boca hay algo, por ejemplo un dedo o la hoja de entrenamiento. Corresponde con el requisito R1.2.10.1

Nombre del Caso de Uso
Eliminar Evento Objeto Boca
Descripción
El usuario borra el evento de detección de objeto en la boca. Corresponde con el requisito R1.2.10.2

Nombre del Caso de Uso
Insertar Evento Variación Luz
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se detecte un cambio significativo en el nivel de la luz. Corresponde con el requisito R1.2.11.1

Nombre del Caso de Uso
Eliminar Evento Variación Luz
Descripción
El usuario borra el evento de variación de luz. Corresponde con el requisito R1.2.11.2

Nombre del Caso de Uso
Insertar Evento Botón Wake Up
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se presione el botón Wake Up, situado cerca de la ranura SD. Corresponde con el requisito R1.2.12.1

Nombre del Caso de Uso
Eliminar Evento Botón Wake Up
Descripción
El usuario borra el evento de pulsación del botón Wake Up. Corresponde con el requisito R1.2.12.2

Nombre del Caso de Uso
Insertar Evento Detección de Pleo
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se detecte la presencia de otro Pleo mediante IR. Corresponde con el requisito R1.2.13.1

Nombre del Caso de Uso
Eliminar Evento Detección de Pleo
Descripción
El usuario borra el evento de detección de otro Pleo. Corresponde con el requisito R1.2.13.2

Nombre del Caso de Uso
Insertar Evento Encendido
Descripción
El usuario inserta un evento que lanzará una serie de acciones cuando se encienda la forma de vida artificial, sin esperar por nada más. Corresponde con el requisito R1.2.14.1

Nombre del Caso de Uso
Eliminar Evento Encendido
Descripción
El usuario borra el evento de encendido. Corresponde con el requisito R1.2.14.2

Nombre del Caso de Uso
Insertar Posición Neutra
Descripción
El usuario inserta la orden de posición neutra, esto configura todas las articulaciones de la forma de vida a su estado neutro, por ejemplo para guardar en la caja. Corresponde con el requisito R1.3.1

Nombre del Caso de Uso
Eliminar Posición Neutra
Descripción
El usuario borra la orden de posición neutra. Corresponde con el requisito R1.3.1

Nombre del Caso de Uso
Insertar Captura de Imagen
Descripción
El usuario inserta la orden de captura de Imagen, le da un nombre y la imagen se guarda en la tarjeta SD. Corresponde con el requisito R1.4.1

Nombre del Caso de Uso
Editar Captura de Imagen
Descripción
El usuario edita la orden de captura de Imagen, cambiando el nombre o el orden. Corresponde con el requisito R1.4.2

Nombre del Caso de Uso
Eliminar Captura de Imagen
Descripción
El usuario borra la orden de captura de Imagen. Corresponde con el requisito R1.4.3

Nombre del Caso de Uso
Insertar Captura de Sonido
Descripción
El usuario inserta la orden de captura de sonido, le da un nombre y duración y el sonido se guarda en la tarjeta SD. Corresponde con el requisito R1.5.1

Nombre del Caso de Uso
Editar Captura de Sonido
Descripción
El usuario edita la orden de captura de sonido, cambiando el nombre o el orden. Corresponde con el requisito R1.5.2

Nombre del Caso de Uso
Eliminar Captura de Sonido
Descripción
El usuario borra la orden de captura de sonido. Corresponde con el requisito R1.5.3

Nombre del Caso de Uso
Insertar Reproducción de Sonido
Descripción
El usuario inserta la orden de reproducción de sonido. Corresponde con el requisito R1.6.1

Nombre del Caso de Uso
Editar Reproducción de Sonido
Descripción
El usuario edita la orden de reproducción de sonido, escogiendo otro. Corresponde con el requisito R1.6.2

Nombre del Caso de Uso
Eliminar Reproducción de Sonido
Descripción
El usuario borra la orden de reproducción de sonido. Corresponde con el requisito R1.6.3

Nombre del Caso de Uso
Insertar Tiempo Espera
Descripción
El usuario inserta la orden de tiempo de espera. Corresponde con el requisito R1.7.1

Nombre del Caso de Uso
Editar Tiempo Espera
Descripción
El usuario edita la orden de tiempo en espera, insertando otro tiempo. Corresponde con el requisito R1.7.2

Nombre del Caso de Uso
Eliminar Tiempo Espera
Descripción
El usuario borra la orden de tiempo en espera. Corresponde con el requisito R1.7.3

Nombre del Caso de Uso
Aumentar Turno
Descripción
El usuario aumenta el turno de una orden, se intercambia con la siguiente. Corresponde con el requisito R2.1

Nombre del Caso de Uso
Disminuir Turno
Descripción
El usuario disminuye el turno de una orden, se intercambian con la anterior. Corresponde con el requisito R2.2

Nombre del Caso de Uso
Nueva Personalidad
Descripción
Crea una nueva Personalidad. Corresponde con el requisito R3.1

Nombre del Caso de Uso
Guardar Personalidad
Descripción
Guarda la personalidad actual. Corresponde con el requisito R3.2

Nombre del Caso de Uso
Cargar Personalidad
Descripción
Carga una personalidad guardada previamente. Corresponde con el requisito R3.3

Nombre del Caso de Uso
Compilar
Descripción
Se ejecuta el módulo de compilación. Corresponde con el requisito R4.1

Nombre del Caso de Uso
Indicar Forma de Vida
Descripción
El usuario indica a que forma de vida se va exportar la personalidad. Corresponde con el requisito R4.2

Nombre del Caso de Uso
Indicar Versión
Descripción
El usuario indica a que versión de la forma de vida se va exportar la personalidad. Corresponde con el requisito R4.3

Nombre del Caso de Uso
Indicar Ruta
Descripción
Se indica en que ubicación se guardará el archivo ejecutable resultante de la operación. Corresponde con el requisito R4.4

Nombre del Caso de Uso
Editar Preferencias
Descripción
Permite al usuario editar las preferencias de la herramienta. Corresponde con el requisito R5.1

Nombre del Caso de Uso
Salvar LOG
Descripción
Se guarda el log. Corresponde con el requisito R5.2

Nombre del Caso de Uso
Ruta Log
Descripción
Se edita la ruta del log. Corresponde con el requisito R5.3

Nombre del Caso de Uso
Mostrar Ayuda
Descripción
Muestra la ayuda del programa. Corresponde con el requisito R6

5.3 Identificación de los Subsistemas en la Fase de Análisis

5.3.1 Descripción de los Subsistemas

El proyecto está compuesto por varios subsistemas que trabajan entre sí.

5.3.1.1 Subsistema Interfaz

El subsistema interfaz es el encargado de facilitar las herramientas necesarias para la creación y edición del diagrama así como mostrar dichas operaciones visualmente en la herramienta. Es el subsistema que interactúa con el usuario a través del DSL. El diagrama debe ajustarse a lo que el usuario ha querido representar y ser una primera barrera para evitar errores en la formación de secuencias así como para evitar ambigüedades. Lleva la gestión de todas las ventanas que se muestran y su lógica.

5.3.1.2 Subsistema Gestor Personalidad

Una vez terminado el diagrama el gestor de personalidad transformará dicho diagrama en un fichero que servirá de entrada al compilador. Este fichero reflejará en su totalidad el diagrama realizado y consecuentemente las órdenes del usuario.

5.3.1.3 Subsistema Compilador

El subsistema compilador será el encargado de transformar el archivo generado por el gestor de secuencias en un archivo ejecutable por la forma de vida artificial elegida, inicialmente Pleo. El compilador tiene que llamar al compilador de la forma de vida artificial elegida pasándole un fichero que sea capaz de interpretar, para ello este subsistema deberá adecuarse a la forma de vida artificial para la que se está programando. Tendrá en cuenta también posibles versiones de cada forma de vida y además podrá mover el archivo resultante directamente a la tarjeta SD si el usuario así lo indica.

5.3.2 Descripción de los Interfaces entre Subsistemas

El subsistema interfaz y gestor de personalidades se encuentran dentro de la misma API y la forma de comunicarse será mediante invocaciones a métodos entre las clases de dichos subsistemas.

El gestor de personalidad se comunica con el compilador ejecutándolo y pasándole como argumento el archivo XML resultante de la interacción del usuario con la interfaz.

5.4 Diagrama de Clases Preliminar del Análisis

5.4.1 Diagrama de Clases

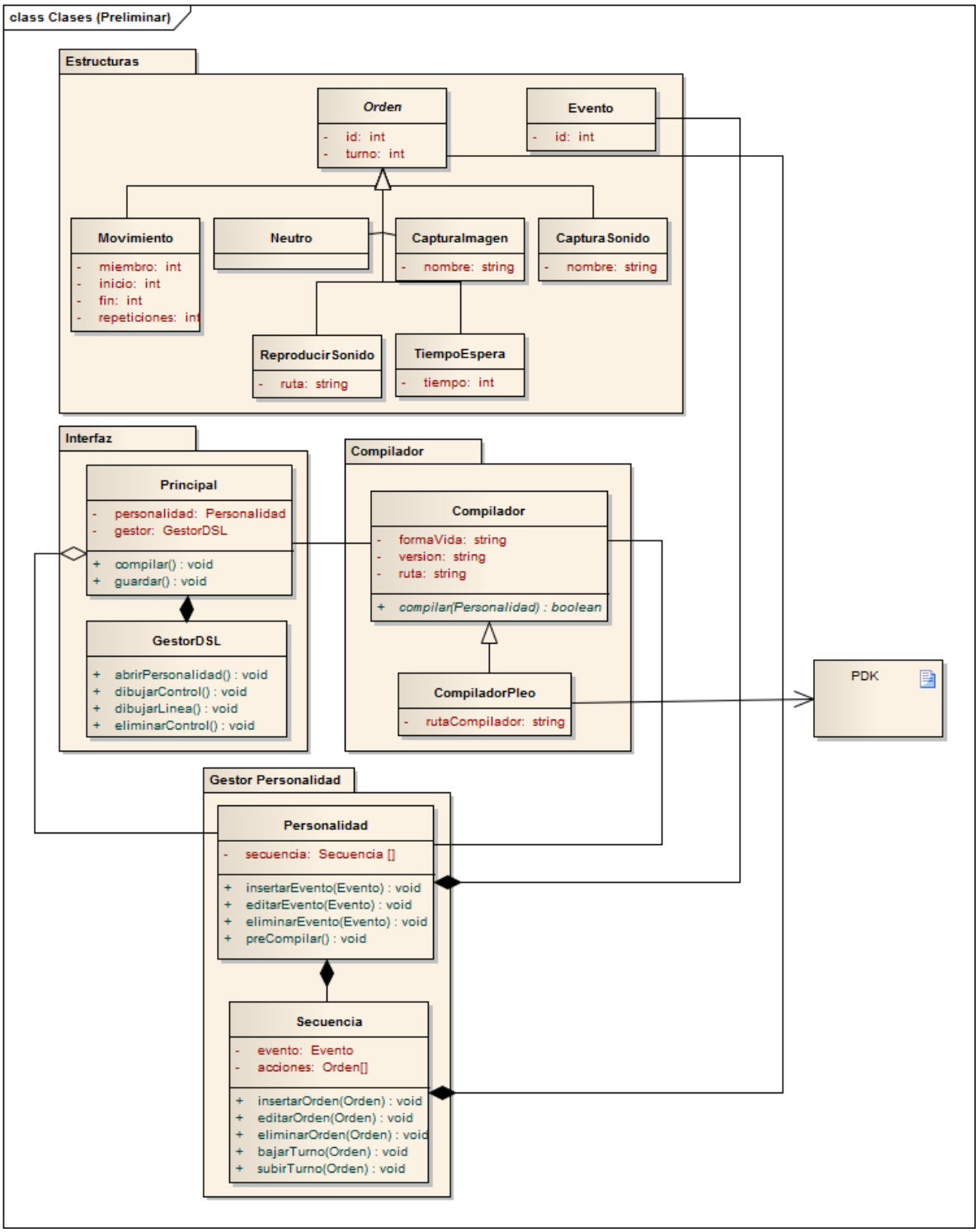


Figura 5.17 Diagrama de Clases Preliminar

5.4.2 Descripción de las Clases

A continuación se detallan las clases que conformarán cada subsistema.

5.4.2.1 Interfaz

El paquete Interfaz es, en principio, el encargado de todo lo relacionado con la parte gráfica de la aplicación. Además será el hilo que una toda la aplicación.

Aquí se encuentra la clase Principal, que se ejecutará la primera y estará en memoria todo el tiempo.

Nombre de la Clase
Principal
Descripción
Clase Principal, ejecuta la aplicación.
Responsabilidades
Administrar los recursos principales de la aplicación y conectarlos. Relacionado con todos los requisitos.
Atributos Propuestos
personalidad: Personalidad actual sobre la que se está trabajando. **gestor**: Gestor de personalidad encargado de realizar operaciones sobre la personalidad y guardar ciertos datos sobre esta.
Métodos Propuestos
compilar: Encargado de llamar a los métodos pertinentes para iniciar el proceso de compilado y llamar al compilador. **guardar**: Guarda el proyecto actual en un archivo que pueda ser abierto posteriormente para seguir editando la personalidad.

Nombre de la Clase
GestorDSL
Descripción
Clase encargada de realizar el dibujado de los componentes gráficos así como de gestionar el DSL, añadiendo secuencias a la personalidad. Abre también, para la edición, proyectos anteriores. Relacionado con R3
Responsabilidades
Gestionar la parte gráfica de la aplicación. Encargado de mantener coherencia en el diagrama DSL.
Métodos Propuestos
abrirPersonalidad: Abre un proyecto anterior. **dibujarControl**: Dibuja los controles que representan las acciones y eventos de la personalidad. **dibujarLinea:** Dibuja las líneas entre los controles. **eliminarControl**: Elimina líneas y controles.

5.4.2.2 Gestor Personalidad

El paquete Gestor Personalidad realiza todas las operaciones necesarias para guardar la personalidad actual (con sus ordenes y eventos) y su gestión, solo el nivel lógico. Es también quien crea el precompilado de la personalidad en un archivo que se pasará como entrada al compilador.

Nombre de la Clase
Personalidad
Descripción
Clase que gestiona la personalidad que se está editando actualmente
Responsabilidades
Mantener en buen estado los atributos de la personalidad para que sean exactos y coherentes con los deseos del usuario. Relacionada con requisitos R1.2 y R4.1
Atributos Propuestos
secuencia: Array con objetos Secuencia.
Métodos Propuestos
insertarEvento: Inserta un nuevo evento, creando una secuencia nueva. **editarEvento**: Edita un evento. **eliminarEvento**: Elimina un evento y su secuencia de acciones. **preCompilar**: Crea un archivo XML a partir de la secuencia para su posterior compilado.

Nombre de la Clase
Secuencia
Descripción
Almacena las ordenes que se hayan de realizar cuando el evento de la secuencia se lance.
Responsabilidades
Mantener en buen estado las órdenes de la secuencia para que sean exactos y coherentes con los deseos del usuario. Relacionada con requisitos R1.1, R1.3, R1.4, R1.5, R1.6, R1.7, R1.8, R2.1, R2.2
Atributos Propuestos
evento: Desencadenante de la secuencia de acciones a realizar. **acciones**: Array de órdenes a realizar.
Métodos Propuestos
insertaOrden: Inserta una nueva orden en la secuencia. **editarOrden: E**dita una orden que se encuentra en la secuencia. **eliminarOrden:** Elimina una orden de la secuencia. **bajarTurno:** Baja el turno de una orden sustituyéndola por la anterior para que se ejecute antes. **subirTurno:** Sube el turno de una orden sustituyéndola por la siguiente para que se ejecute después.

5.4.2.3 Compilador

En el subsistema compilador están las clases de compilado de cada una de las formas de vida para las que se soporte la programación. En un principio solo para la forma de vida Pleo.

Nombre de la Clase
Compilador
Descripción
Realiza la labor del compilado dependiendo de diversas opciones
Responsabilidades
Compilar según las opciones que se le pasen la personalidad en un archivo ejecutable por la forma de vida. Relacionada con los requisitos R4
Atributos Propuestos
formaVida: Forma de vida para la que se realizará el compilado **version:** Versión de la forma de vida. **ruta: R**uta donde se almacenará el resultado de la compilación.
Métodos Propuestos
compilar: Compilado de la personalidad.

Nombre de la Clase
CompiladorPleo
Descripción
Implementa el método compilar con las características de la forma de vida Pleo
Responsabilidades
Compilar para la forma de vida Pleo un archivo ejecutable que represente fielmente las órdenes programadas en el diagrama-DSL. Relacionada con requisitos R4
Atributos Propuestos
rutaCompilador: ruta del compilador PDK
Métodos Propuestos
compilar: Compilado de la personalidad.

5.4.2.4 Estructuras

El paquete estructuras no pertenece a ningún subsistema propiamente dicho, aunque en mayor parte es utilizado por el gestor de Personalidad se utiliza en otros. Alberga las estructuras de datos necesarias para guardar las órdenes y eventos.

Nombre de la Clase
Orden
Descripción
Clase de la que heredan las ordenes para implementar sus propios atributos
Responsabilidades
Relacionada con requisitos R1.1, R1.3, R1.4, R1.5, R1.6, R1.7, R1.8
Atributos Propuestos
id: identificación de la orden. **turno:** turno de la orden dentro de la secuencia.
Métodos Propuestos
get y set de id y turno.

Nombre de la Clase
Movimiento
Descripción
Guarda la orden de movimiento
Responsabilidades
La orden debe encontrarse dentro de los parámetros admitidos. Relacionada con requisitos R1.1
Atributos Propuestos
miembro: Miembro o extremidad que va a moverse **inicio:** Grados de inicio si fuera un movimiento repetitivo **fin:** Grados de posición **repeticiones:** número de repeticiones del movimiento
Métodos Propuestos
gets y sets de los atributos

Nombre de la Clase
Neutro
Descripción
Guarda la orden de posición neutra.
Responsabilidades
Relacionada con requisitos R1.3
Atributos Propuestos
Métodos Propuestos

Nombre de la Clase
CapturaImagen
Descripción
Guarda la orden de captura de imagen.
Responsabilidades
Relacionada con los requisitos R1.4
Atributos Propuestos
nombre: nombre con el que guardará la imagen realizada.
Métodos Propuestos
get y set de nombre

Nombre de la Clase
CapturaSonido
Descripción
Guarda la orden de captura de sonido
Responsabilidades
Relacionada con los requisitos R1.5
Atributos Propuestos
nombre: nombre con el que se guardará el archivo de sonido**.**
Métodos Propuestos
get y set de nombre

Nombre de la Clase
ReproducirSonido
Descripción
Guarda la orden de reproducción de sonido.
Responsabilidades
Relacionada con los requisitos R1.6, R1.7
Atributos Propuestos
ruta: ruta del archivo a reproducir
Métodos Propuestos
get y set de ruta

Nombre de la Clase
TiempoEspera
Descripción
Guarda la orden de tiempo en espera.
Responsabilidades
Relacionada con los requisitos R1.8
Atributos Propuestos
tiempo: tiempo en milisegundos de espera.
Métodos Propuestos
get y set de tiempo.

Nombre de la Clase
Evento
Descripción
Guarda un evento.
Responsabilidades
Relacionada con los requisitos R1.2
Atributos Propuestos
id: identificador que sirve para saber el tipo de evento
Métodos Propuestos
get y set de id.

5.5 Análisis de Casos de Uso y Escenarios

Se presentan a continuación los escenarios más comunes de la aplicación.

5.5.1 Insertar evento

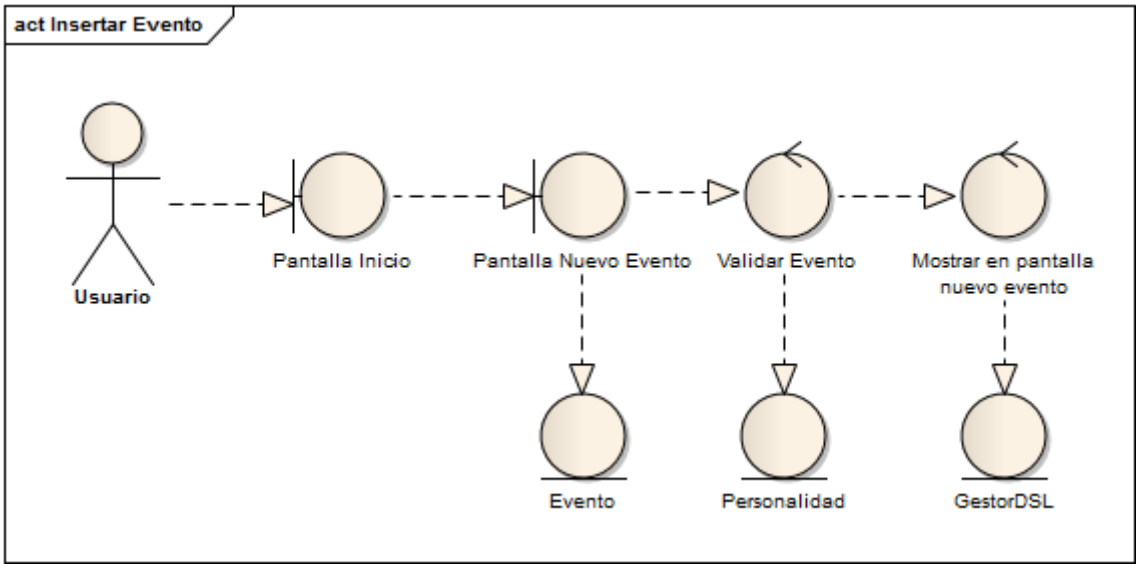


Figura 5.18 Diagrama de robustez: insertar evento

Insertar evento	
Precondiciones	Aplicación abierta y a la espera para introducir una nueva orden o evento.
Poscondiciones	Se crea un nuevo evento.
Actores	Usuario del sistema
Descripción	El usuario: 1. Abre la aplicación. 2. Insertar un nuevo evento. 3. Se valida el evento. 4. Se inserta a la personalidad 5. Se muestra el diagrama modificado.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Ya existe el evento que se ha intentado insertar. ○ Notificar al usuario. ○ Vuelta al paso 2. Se elige un evento distinto.
Notas	Hay cerca de 20 eventos que no necesitan de argumentos para ser insertados. Este diagrama se corresponde con los requisitos R1.2

5.5.2 Eliminar evento

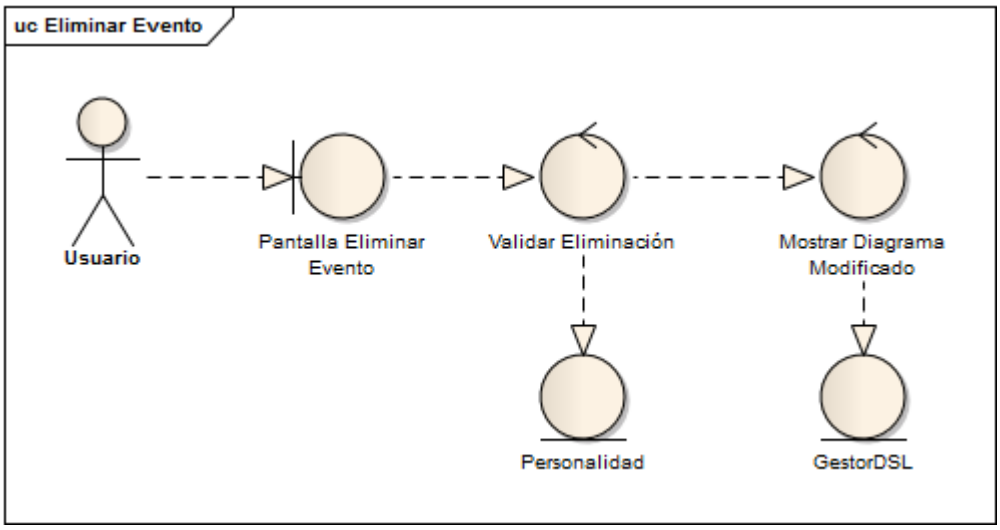


Figura 5.19 Diagrama de robustez: eliminar evento

Eliminar evento	
Precondiciones	Aplicación abierta y con algún evento que eliminar.
Poscondiciones	Se elimina un evento
Actores	Usuario del sistema
Descripción	El usuario: 1. Elige el evento a eliminar. 2. Lo selecciona y elimina 3. Se valida la eliminación. 4. Se muestra el diagrama modificado.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El evento a eliminar tiene órdenes. ○ Notificar al usuario. ○ Se le dan las siguientes opciones. ▪ Eliminar por completo la secuencia de evento y órdenes. ▪ Cancelar la eliminación.
Notas	Este diagrama se corresponde con los requisitos R1.2

5.5.3 Insertar orden

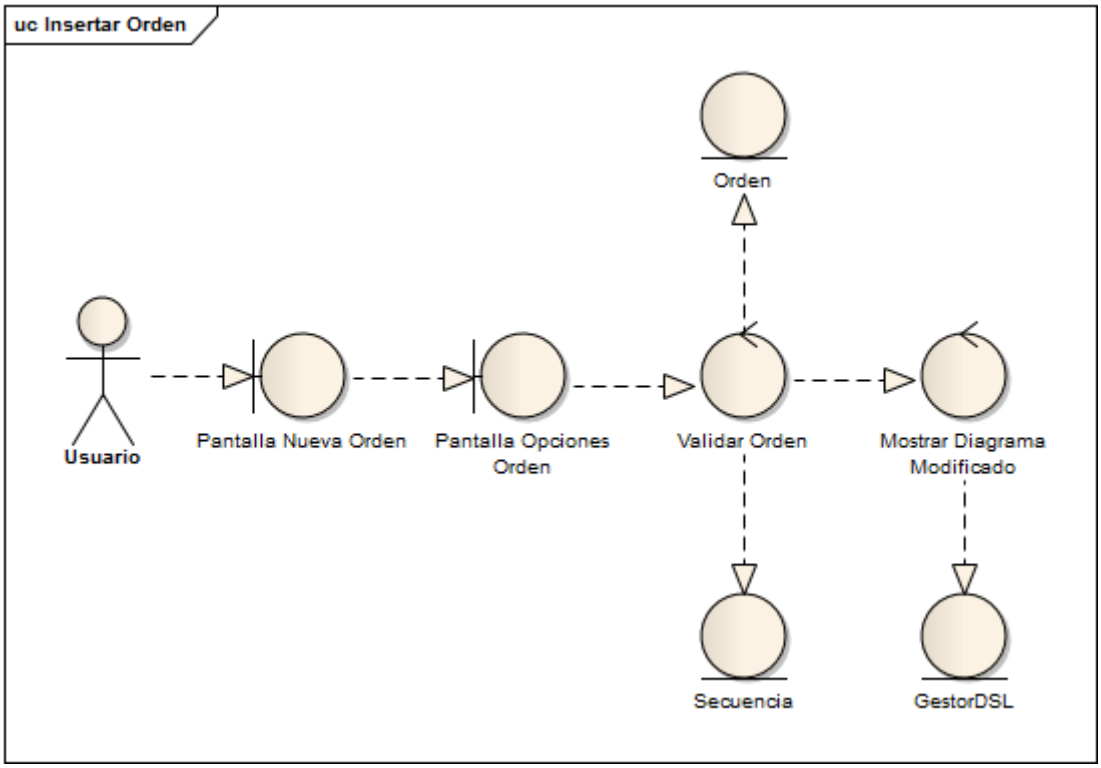


Figura 5.20 Diagrama de robustez: insertar orden

Insertar orden	
Precondiciones	Es necesario que el usuario haya insertado un evento que desencadene la ejecución de la orden que se va a insertar. Puede ser que haya más órdenes ya insertadas para dicho evento.
Poscondiciones	Una nueva orden se ha insertado a la secuencia de un evento.
Actores	Usuario del sistema
Descripción	El usuario: 1. Elige una orden para insertar 2. Se muestra una pantalla con las posibles opciones y argumentos necesarios para configurar la orden. 3. Se valida la orden. 4. Se inserta a la secuencia. 5. Se muestra el diagrama modificado.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Los argumentos de la orden son incorrectos. ○ Notificar al usuario. ○ Vuelta al paso 2.
Notas	Cada orden requiere unos argumentos determinados. Estos deben ser validos para cada caso. Se corresponde con los requisitos R1.1, R1.3, R1.4, R1.5, R1.6, R1.7, R1.8

5.5.4 Editar orden

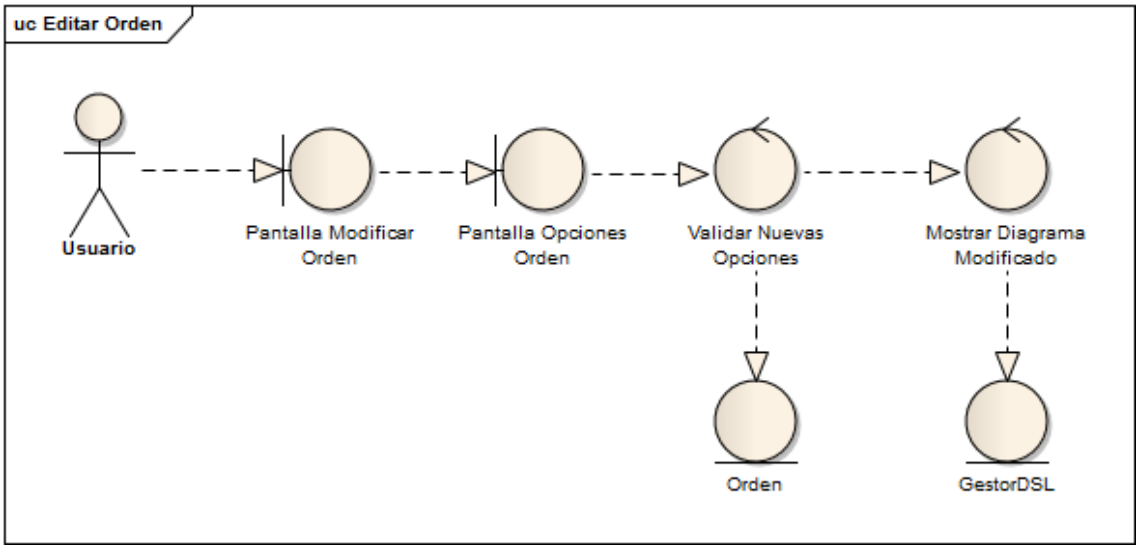


Figura 5.21 Diagrama de robustez: editar orden

Editar orden	
Precondiciones	Es necesario que el usuario haya insertado un evento con por lo menos una orden.
Poscondiciones	Una orden existente se ha modificado.
Actores	Usuario del sistema
Descripción	El usuario: 1. Elige una orden para modificar 2. Se muestra una pantalla con las posibles opciones y argumentos necesarios para configurar la orden. 3. Se valida la orden. 4. Se muestra el diagrama modificado.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Los argumentos de la orden son incorrectos. ○ Notificar al usuario. ○ Vuelta al paso 2.
Notas	Cada orden requiere unos argumentos determinados. Estos deben ser validos para cada caso. Se corresponde con los requisitos R1.1, R1.3, R1.4, R1.5, R1.6, R1.7, R1.8

5.5.5 Eliminar orden

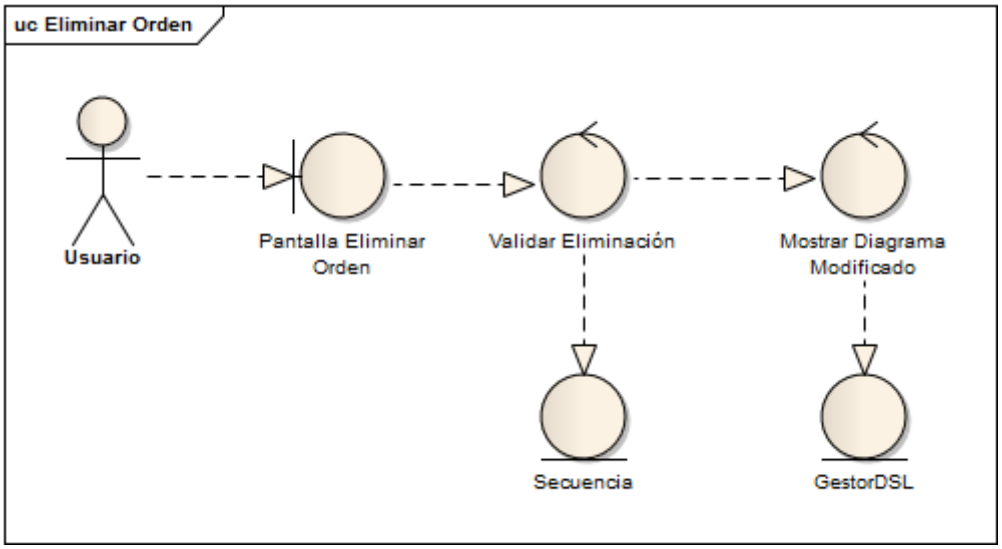


Figura 5.22 Diagrama de robustez: eliminar orden

Eliminar orden	
Precondiciones	Es necesario que el usuario haya insertado un evento con por lo menos una orden.
Poscondiciones	Una orden existente se ha borrado.
Actores	Usuario del sistema
Descripción	El usuario: 1. Elige una orden para eliminar. 2. Se valida la eliminación. 3. Se muestra el diagrama modificado.
Notas	Se corresponde con los requisitos R1.1, R1.3, R1.4, R1.5, R1.6, R1.7, R1.8

5.5.6 Aumentar turno orden

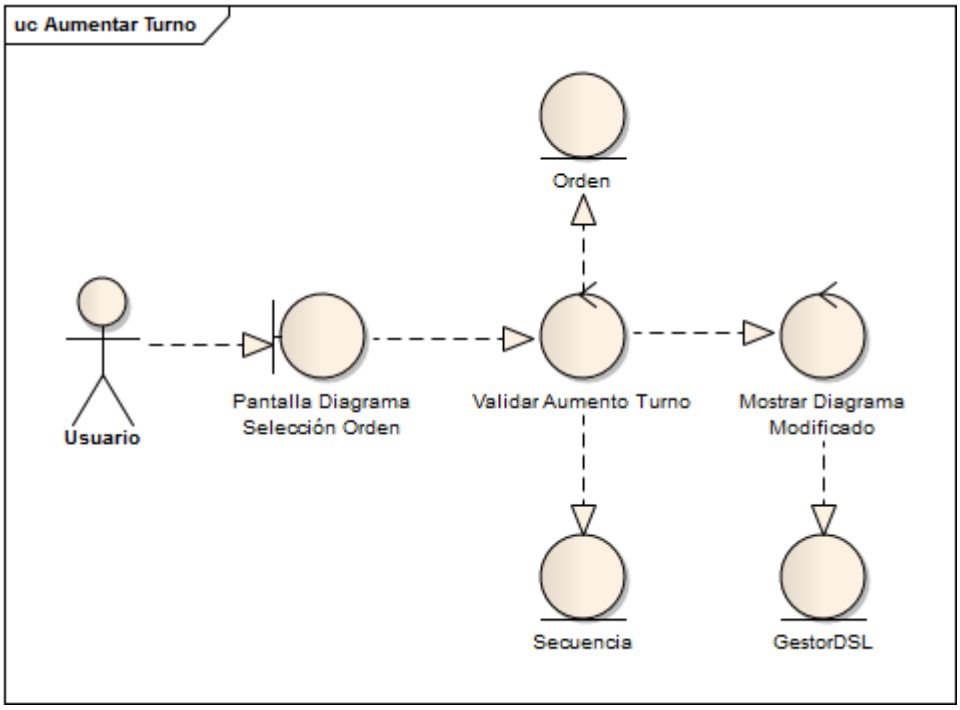


Figura 5.23 Diagrama de robustez: aumentar turno orden

Aumentar turno orden	
Precondiciones	Es necesario que el usuario haya insertado un evento con por lo menos dos órdenes.
Poscondiciones	El turno de una orden existente se ha modificado
Actores	Usuario del sistema
Descripción	El usuario: 1. Elige una orden para aumentar su turno. 2. Se valida el aumento de turno. 3. Se muestra el diagrama modificado.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: La orden seleccionada es la última en el turno. ○ Notificar al usuario. ○ La orden no sufre cambios
Notas	Se corresponde con el requisito R2.1

5.5.7 Disminuir turno orden

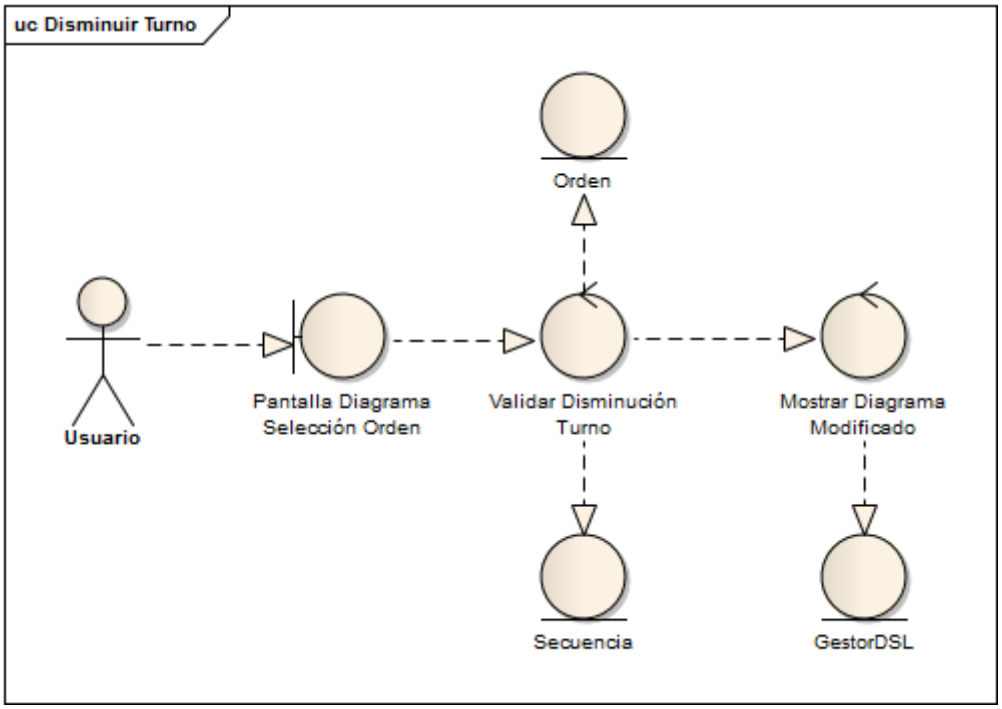


Figura 5.24 Diagrama de robustez: disminuir turno orden

Disminuir turno orden	
Precondiciones	Es necesario que el usuario haya insertado un evento con por lo menos dos órdenes.
Poscondiciones	El turno de una orden existente se ha modificado.
Actores	Usuario del sistema
Descripción	El usuario: 1. Elige una orden para disminuir su turno. 2. Se valida la disminución de turno. 3. Se muestra el diagrama modificado.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: La orden seleccionada es la primera en el turno. ○ Notificar al usuario. ○ La orden no sufre cambios
Notas	Se corresponde con el requisito R2.2

5.5.8 Compilar

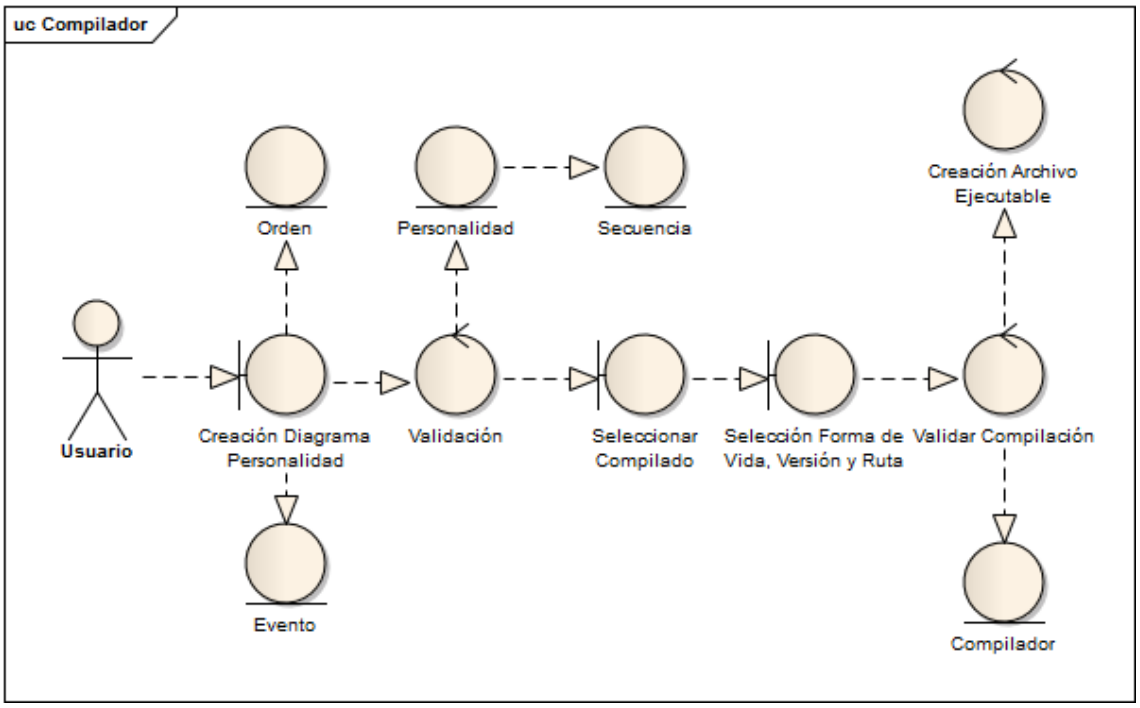


Figura 5.25 Diagrama de robustez: compilar

Compilar	
Precondiciones	Aplicación abierta.
Poscondiciones	Se crear un archivo ejecutable por la forma de vida artificial.
Actores	Usuario del sistema
Descripción	El usuario: 1. Crea una nueva personalidad. 2. Se selecciona el compilado. 3. Se elige la forma de vida, versión y ruta del archivo que se creará. 4. Se valida la compilación y se crea un archivo ejecutable.
Excepciones	• **El compilador de la forma de vida da un error desconocido.** El compilador no crea el archivo ejecutable debido a un error en la compilación**.** ○ Se notifica al usuario el error. ○ Se presentan posibles causas. ○ Se presentan posibles soluciones.
Notas	Este diagrama tiene la interfaz “Creación Diagrama Personalidad” que se trata de un proceso de insertar eventos y órdenes ya descrito anteriormente. Este diagrama se corresponde con los requisitos R4

5.6 Análisis de Interfaces de Usuario

5.6.1 Descripción de la Interfaz

A continuación se muestra un boceto de la interfaz de la ventana principal. Esta organizado en 4 partes que se describen con más detalle a continuación.

Figura 5.26 Boceto de la interfaz principal

- **Menú**. En esta parte se colocará el típico menú de aplicación, para realizar operaciones comunes como abrir, guardar, cerrar, ayuda, etc.
- **Herramientas**. En esta zona se colocarán todos los botones y controles necesarios para insertar, editar y eliminar los eventos y órdenes. En principio se elegirá el evento u orden deseado y se pinchará en la zona del diagrama donde se desee colocar.
- **Zona Diagrama DSL**. Es la parte principal de la interfaz y donde se creará el diagrama que represente la personalidad de la forma de vida artificial.
- **Barra de estado**. Mostrará un mensaje con el estado actual de la aplicación.

Cuando se inserte un evento u orden en el diagrama es posible que requiera de opciones adicionales para que esté completamente detallado. En ese caso aparecerá una ventana con las opciones pertinentes para cada caso.

A continuación se muestra un ejemplo de ventana que se mostrará al insertar una orden de movimiento.

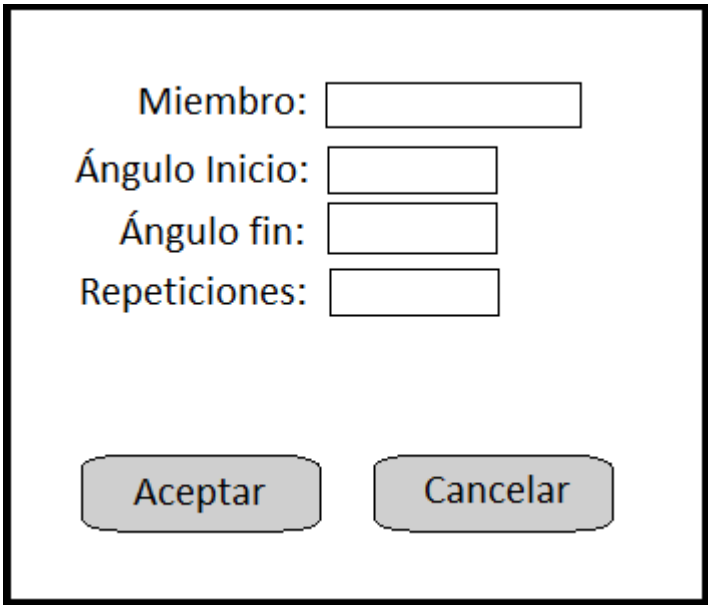


Figura 5.27 Boceto de la ventana de nueva orden de movimiento

En este ejemplo el campo miembro sirve para elegir de una lista desplegable el miembro a mover. El ángulo de fin donde se situará el miembro y el número de repeticiones, que encaso de ser superior a 1 será necesario cubrir también el campo ángulo inicio.

En el siguiente boceto se muestra la ventana de compilación y las opciones que se mostrarán para dicho proceso, será necesario cubrir todas.

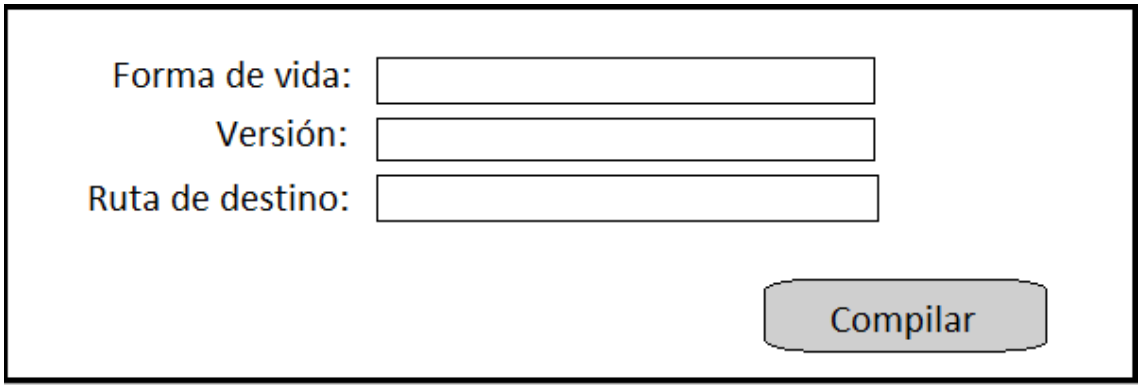


Figura 5.28 Boceto de la ventana de compilado

5.6.2 Descripción del Comportamiento de la Interfaz

Las entradas que realice el usuario serán en todo caso verificadas para que se controle la correcta entrada de los datos. Algunos ejemplos son la inserción de opciones para nuevas órdenes o eventos. En estos casos se notificará al usuario mediante un diálogo que ha introducido un dato erróneo. Como los datos correctos introducidos para una orden pueden ser erróneos para otra, se comprobará en el momento de dar a aceptar la validez de dichos datos.

Se incluirá una ayuda para el uso de la aplicación así como otros documentos y anexos que se consideren de importancia para el desarrollo en cada forma de vida. Estos documentos podrán ser consultados en el momento por el usuario para verificar las órdenes y los parámetros que correctos en cada caso.

5.6.3 Diagrama de Navegabilidad

El siguiente diagrama de navegabilidad representa la posible navegación entre las pantallas de la herramienta. Desde los objetos control se vuelve de nuevo a la pantalla principal, esta relación no se ha representado para simplificar el diagrama.

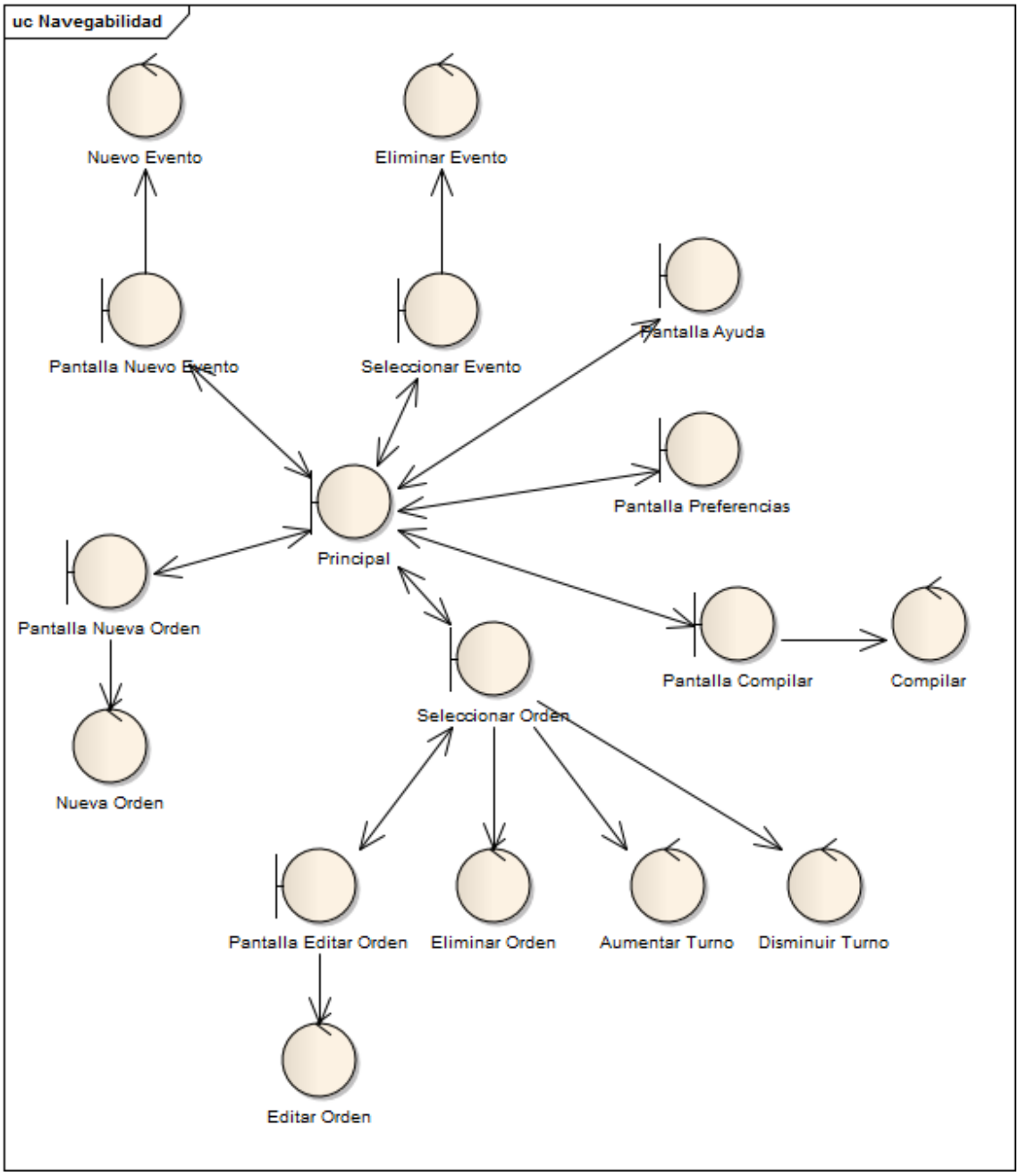


Figura 5.29 Diagrama de Navegabilidad

5.7 Especificación del Plan de Pruebas

5.7.1 Pruebas unitarias

Caso de Uso 1: Insertar Evento	
Prueba	**Resultado Esperado**
Guardar un evento	Se guarda correctamente un evento nuevo.
Prueba	**Resultado Esperado**
Cancelar la Operación	No se guarda ningún evento.

Caso de Uso 2:Eliminar Evento	
Prueba	**Resultado Esperado**
Eliminar un evento.	Se elimina correctamente un evento.
Prueba	**Resultado Esperado**
Cancelar la Operación	No se elimina ningún evento.

Caso de Uso 3: Insertar Orden	
Prueba	**Resultado Esperado**
Guardar una orden	Se guarda correctamente una orden nueva.
Prueba	**Resultado Esperado**
Cancelar la Operación	No se guarda ninguna orden.

Caso de Uso 4: Modificar Orden	
Prueba	**Resultado Esperado**
Se modifica una orden	Se modifican los datos de una orden.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 5: Eliminar Orden	
Prueba	**Resultado Esperado**
Eliminar una orden	Se elimina una orden.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 6: Aumentar Turno	
Prueba	**Resultado Esperado**
Aumentar turno de una orden.	La orden tiene un turno mayor.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 7: Disminuir Turno	
Prueba	**Resultado Esperado**
Disminuir turno de una orden	La orden tiene un turno menor.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

5.7.2 Pruebas de Integración y del Sistema

Caso de Uso 1: Insertar Evento	
Prueba	**Resultado Esperado**
Insertar un evento	El sistema posee un evento más y se dibuja en el diagrama.
Prueba	**Resultado Esperado**
Insertar un evento repetido	El sistema no posee un evento más y se muestra un dialogo notificándolo
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 2:Eliminar Evento	
Prueba	**Resultado Esperado**
Eliminar un evento sin órdenes.	El sistema tiene un evento menos y se borra del diagrama.
Prueba	**Resultado Esperado**
Eliminar un evento con órdenes.	El sistema muestra un diálogo confirmando que se desean eliminar todas las órdenes de dicho evento y el sistema tiene un evento menos y todas las ordenes que este tuviera. Se borra el evento y las órdenes del diagrama.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 3: Insertar Orden	
Prueba	**Resultado Esperado**
Insertar una orden existiendo un evento	El sistema posee un evento con una orden más y se dibuja en el diagrama.
Prueba	**Resultado Esperado**
Insertar una orden sin que exista ningún evento.	El sistema no posee una orden más y se muestra un dialogo notificándolo.
Prueba	**Resultado Esperado**
Insertar una orden con opciones incorrectas.	El sistema no posee una orden más y se muestra un dialogo notificando el error.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 4: Modificar Orden	
Prueba	**Resultado Esperado**
Modificar una orden con opciones correctas.	El sistema posee la misma orden pero con las nuevas opciones y se modifica el control en el diagrama
Prueba	**Resultado Esperado**
Modificar una orden con opciones incorrectas	El sistema no modifica la orden y se muestra un diálogo notificándolo.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 5: Eliminar Orden	
Prueba	**Resultado Esperado**
Eliminar una orden existente	El sistema posee una orden menos y se elimina del diagrama.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 6: Aumentar Turno	
Prueba	**Resultado Esperado**
Se aumenta el turno de una orden que no esté en último lugar.	El sistema posee la misma orden pero intercambiada en el turno con la siguiente y se actualiza el diagrama.
Prueba	**Resultado Esperado**
Se aumenta el turno de una orden que se encuentra en último lugar.	El sistema no realiza ninguna modificación.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 7: Disminuir Turno	
Prueba	**Resultado Esperado**
Se disminuye el turno de una orden que no esté en primer lugar.	El sistema posee la misma orden pero intercambiada en el turno con la anterior y se actualiza el diagrama.
Prueba	**Resultado Esperado**
Se disminuye el turno de una orden que se encuentra en primer lugar.	El sistema no realiza ninguna modificación.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 8: Compilar	
Prueba	**Resultado Esperado**
Compilar un programa válido.	El sistema crea un nuevo archivo ejecutable.
Prueba	**Resultado Esperado**
Compilar un programa no válido.	El sistema no posee un nuevo archivo ejecutable y se muestra un diálogo notificándolo.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

5.7.3 Pruebas de usabilidad

Se deberá verificar que la aplicación es suficiente sencilla para cualquier tipo de persona, constatando que es **fácil de aprender** y de **utilizar**. Para ello se hará énfasis en los siguientes aspectos en su construcción:

- **Diseño de las ventanas**. Tener en cuenta los principios de predicción, síntesis, familiaridad y consistencia.
- **Colores**. Combinaciones de colores compatibles, evitar colores brillantes y utilizar códigos redundantes.
- **Shortcuts**. Mostrar los posibles atajos.
- **Tooltips**. Utilizar estos pequeños cuadros de ayuda cuando se pase el ratón por encima de algún elemento.
- **Ayuda**. Incluir en la herramienta una pequeña ayuda que sirva al usuario para orientarse y realizar las operaciones.
- **Documentación**. Acompañar la herramienta de diferentes manuales para la utilización de esta.

Capítulo 6. Diseño del Sistema

6.1 Arquitectura del Sistema

6.1.1 Diagramas de Paquetes

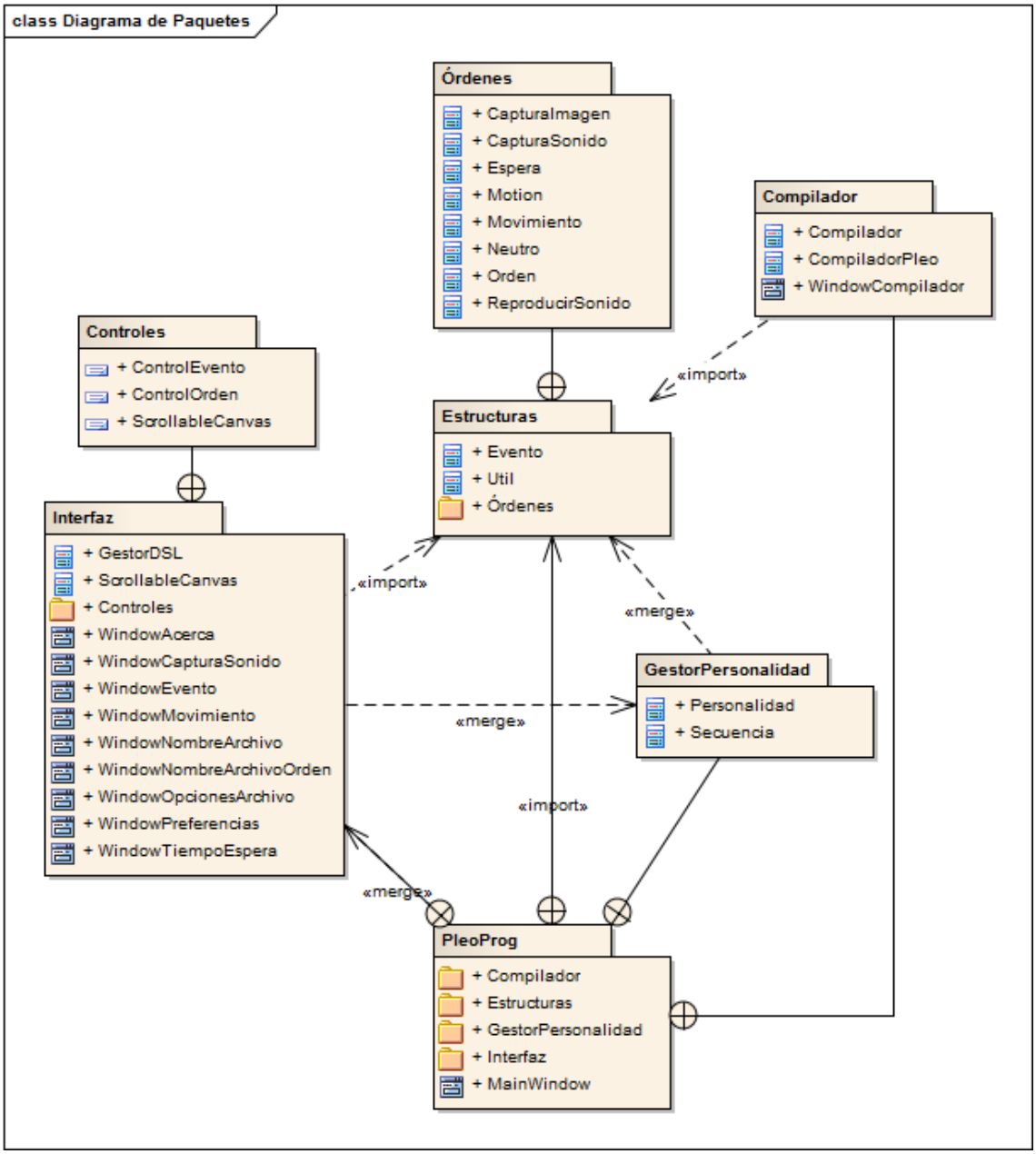


Figura 6.1 Diagrama de paquetes

6.1.1.1 PleoProg

Es el paquete raíz de la aplicación, no es un paquete como tal dentro del proyecto pero se encuentra referenciado y representado para mostrarlo como el nexo de unión entre los diferentes paquetes. Alberga la definición de la ventana principal que se muestra al abrir la aplicación. Esta ventana, presente mientras la herramienta se encuentre activa, guarda en memoria el gestor de todos los elementos gráficos de la aplicación.

Ventanas de usuario que contiene:

- MainWindow

6.1.1.2 Interfaz

El paquete Interfaz contiene la clase creada para la gestión del diagrama a nivel gráfico y todas las ventanas que se mostrarán al usuario para insertar, editar y eliminar órdenes y eventos entre otras acciones. Además contiene el elemento redefinido ScrollableCanvas, para que se ajuste el canvas al proyecto.

La clase que contiene es:

- GestorDSL

Las ventanas de usuario que contiene:

- WindowAcerca
- WindowCapturaSonido
- WindowEvento
- WindowMovimiento
- WindowNombreArchivo
- WindowNombreArchivoOrden
- WindowOpcionesArchivo
- WindowPreferencias
- WindowTiempoEspera

Además contiene un paquete con dos elementos gráficos, controles, creados para trabajar en este entorno y un ScrollableCanvas que hereda de Canvas. Se trata del paquete Controles que contiene los siguientes controles personales:

- ControlEvento
- ControlOrden
- ScrollableCanvas

6.1.1.3 Estructuras

El paquete estructuras tiene las clases que representan los eventos y órdenes. Las clases que contiene son:

- Evento
- Util

Además contiene otro paquete llamado Órdenes con las clases de órdenes:

- CapturaImagen
- CapturaSonido
- Espera
- Motion
- Movimiento
- Neutro
- Orden
- ReproducirSonido

6.1.1.4 GestorPersonalidad

El gestor de personalidad es la parte que contiene la lógica de la personalidad creada, a través de las clases que contiene almacena la personalidad que se esté creando.

Las clases que contiene son:

- Personalidad
- Secuencia

6.1.1.5 Compilador

El compilador contiene las clases específicas para el compilado en las diversas formas de vida así como la pantalla que posibilita esta compilación.

Las clases que lo conforman son:

- Compilador
- CompiladorPleo

La ventana que contiene es:

- WindowCompilador

6.1.2 Diagrama de Componentes

A continuación se muestra el diagrama de componentes que se encuentra dividido en los subsistemas mencionados anteriormente.

La **Interfaz** es la parte encargada de toda la interacción con el usuario así como de mostrar los elementos gráficos de forma que se correspondan con las diferentes operaciones que va realizando el usuario. Es el nexo entre los diferentes subsistemas puesto que es quien ejecuta las diferentes acciones según el usuario lo requiera.

El **GestorPersonalidad** se ocupa de la parte lógica de la personalidad que se está creando, crea y edita la personalidad de la forma de vida artificial según las acciones del usuario y la almacena en memoria mientras sea necesaria. Además provee un XML de la personalidad actual para que sea compilado.

El **Compilador** es el encargado de traducir el fichero XML generado por GestorPersonalidad en un fichero ejecutable por la forma de vida escogida.

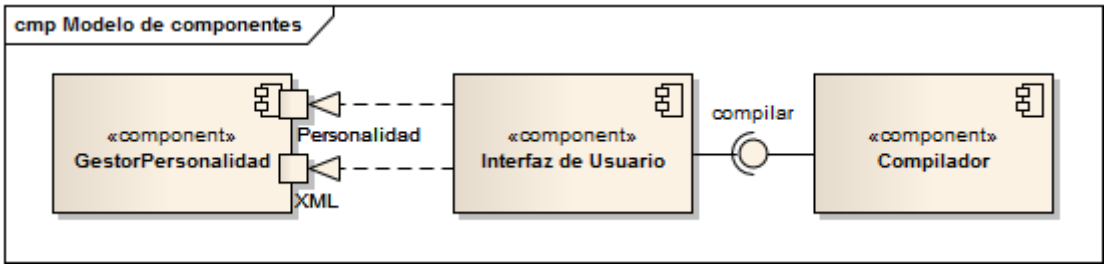


Figura 6.2 Diagrama de componentes

6.2 Diseño de Clases

A continuación se muestran diferentes diagramas de clases que componen la herramienta desarrollada, se han dividido en varios diagramas para su mejor comprensión debido a su extensión. Se muestra un diagrama general en el que se representan todas las clases y sus relaciones. En los demás se obvian las relaciones.

6.2.1 Diagrama de Clases General

El siguiente diagrama representa los paquetes Interfaz, Compilador y GestorPersonalidad a fondo y el paquete Estructuras sin entrar a detalle. También se muestran las clases de la interfaz, que se detallarán en los siguientes diagramas.

Además muestra el uso de CompiladorPleo del PDK necesario.

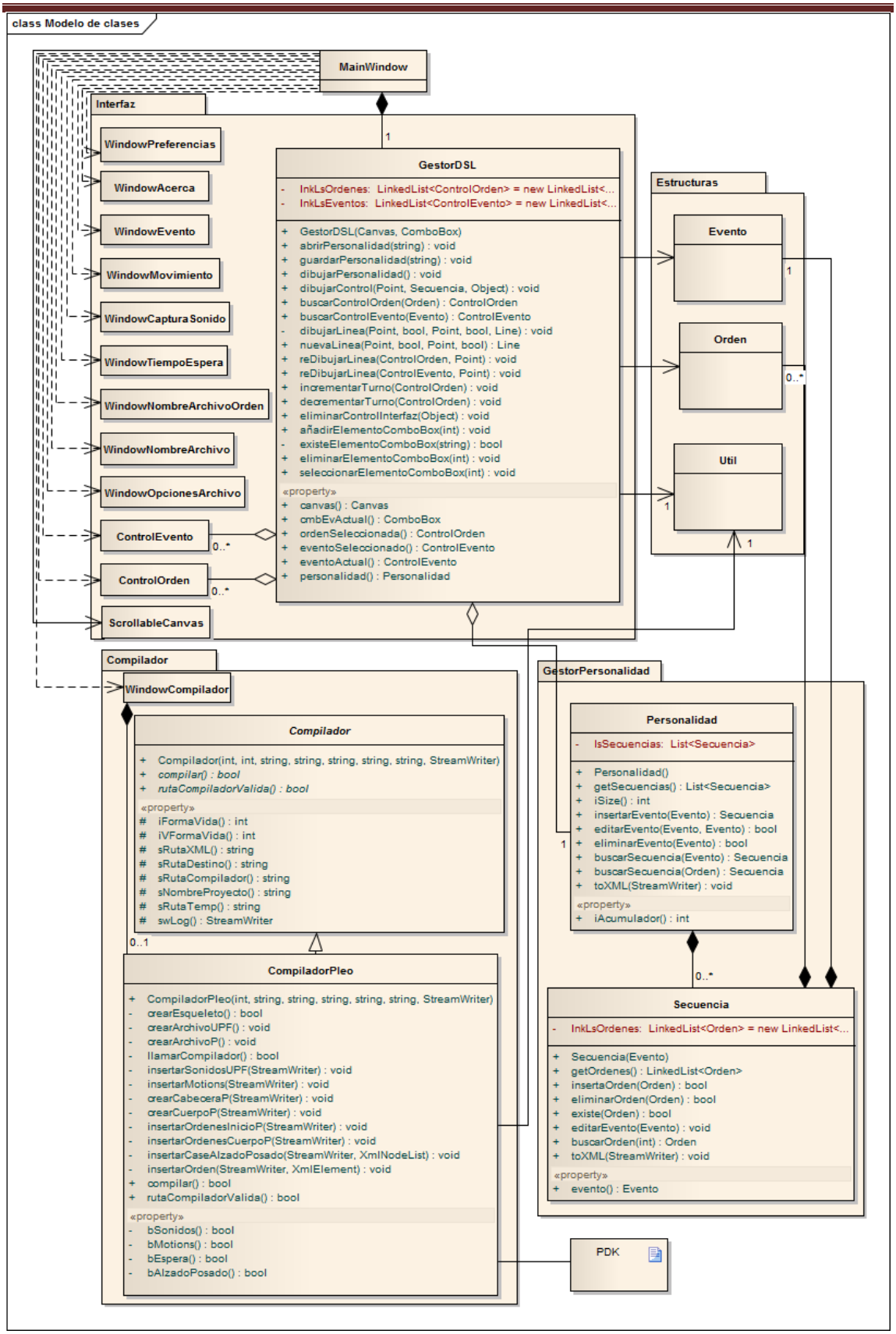


Figura 6.3 Diagrama de clases general

A continuación se muestran las clases Window más en detalle en dos diagramas.

6.2.2 Diagrama de Clases de Ventanas

Figura 6.4 Diagrama de clases ventanas I

A continuación se muestran el resto de clases del paquete interfaz. ControlEvento y ControlOrden se han creado para mostrar la representación gráfica de los eventos y órdenes, respectivamente, dentro del canvas.

El canvas que se utiliza ha sido personalizado y redefinido en ScrollableCanvas para que se pudiera navegar a través de las barras horizontal y vertical.

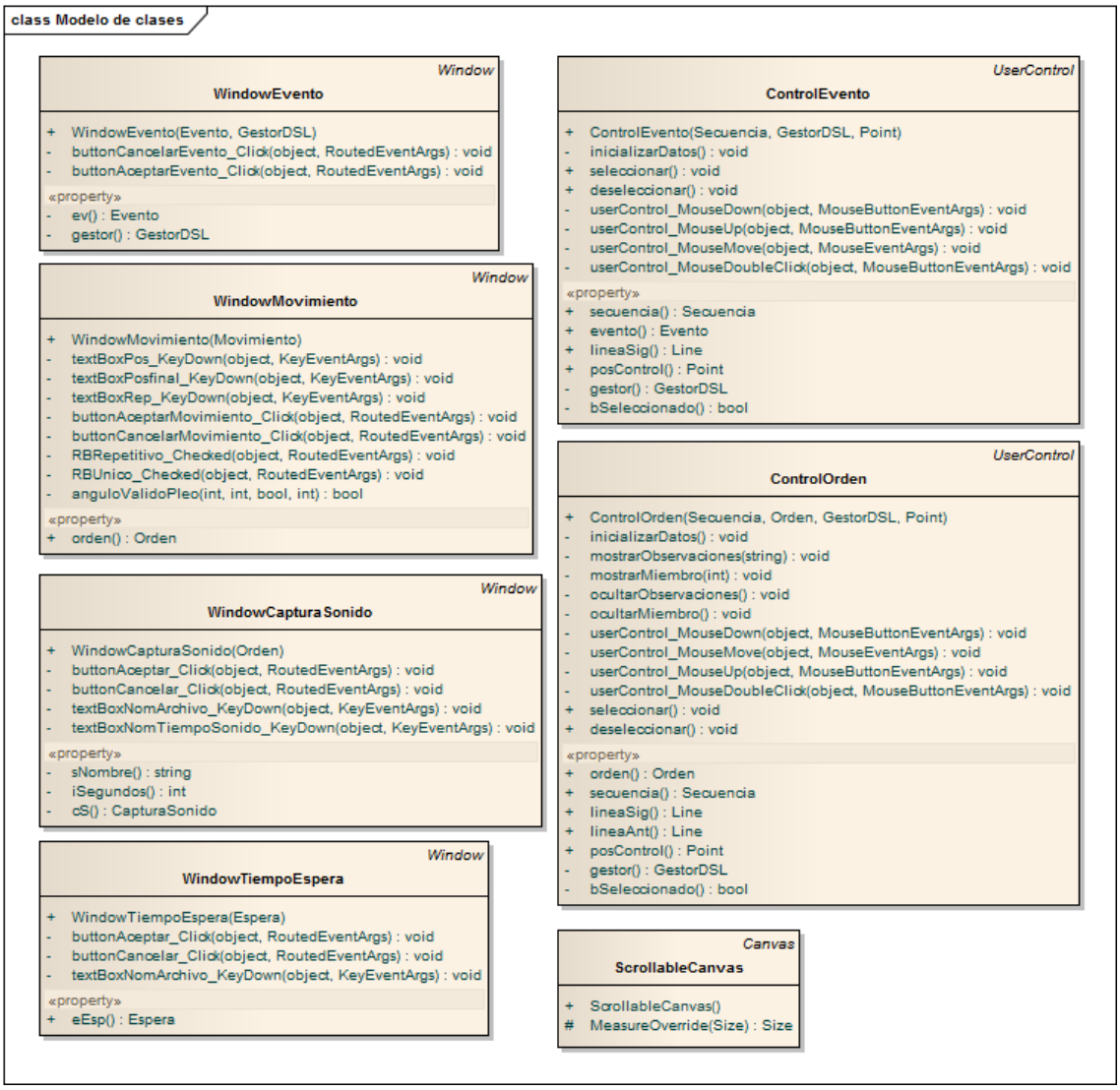


Figura 6.5 Diagrama de clases ventanas II

6.2.3 Diagrama de Clases Paquete Estructura

A continuación se muestra el diagrama de clases del paquete estructuras, que se mostraba simplificado en el anterior apartado. La clase Util se muestra simplificada por su extensión.

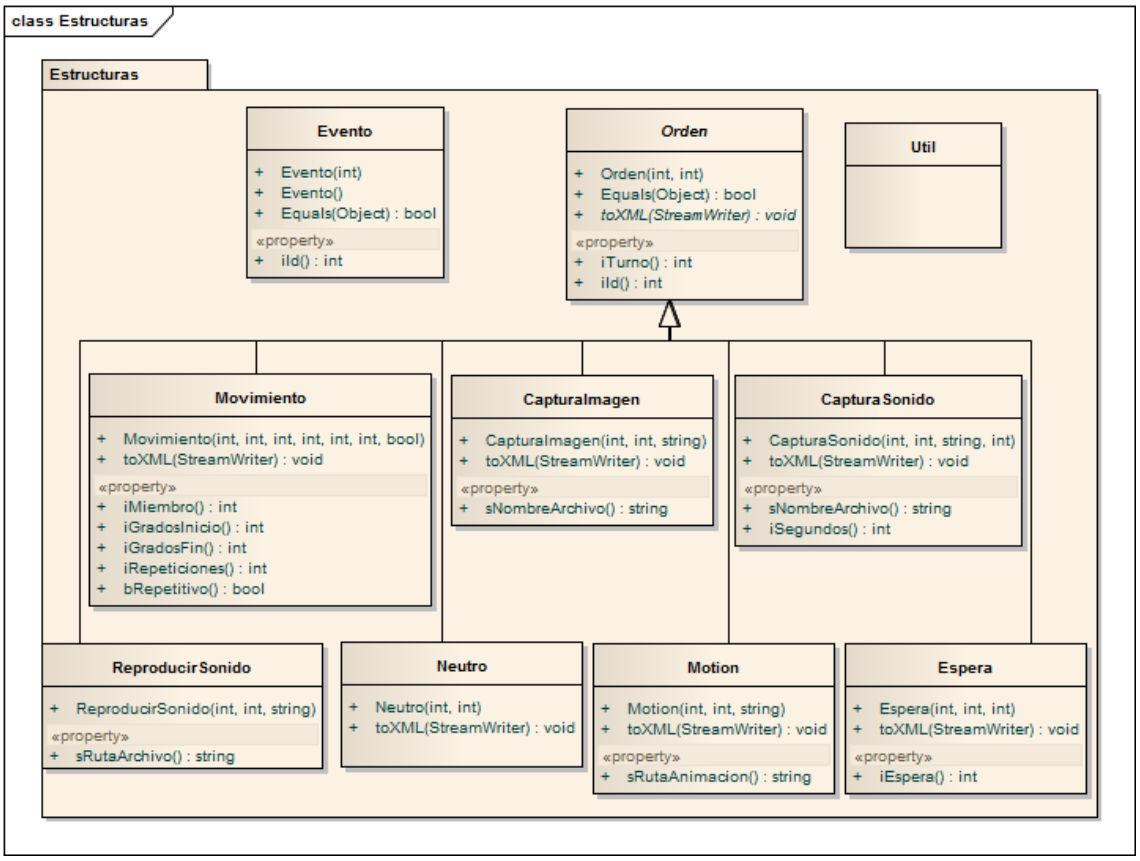


Figura 6.6 Diagrama Estructuras

La clase Util se utiliza desde diferentes clases, como se ha podido ver en el diagrama de clases general anteriormente y contiene variables globales así como métodos que proporcionan nombres y valores asignados a órdenes, eventos y miembros de las formas de vida artificiales.

En la siguiente figura se muestra el contenido de la clase Util.

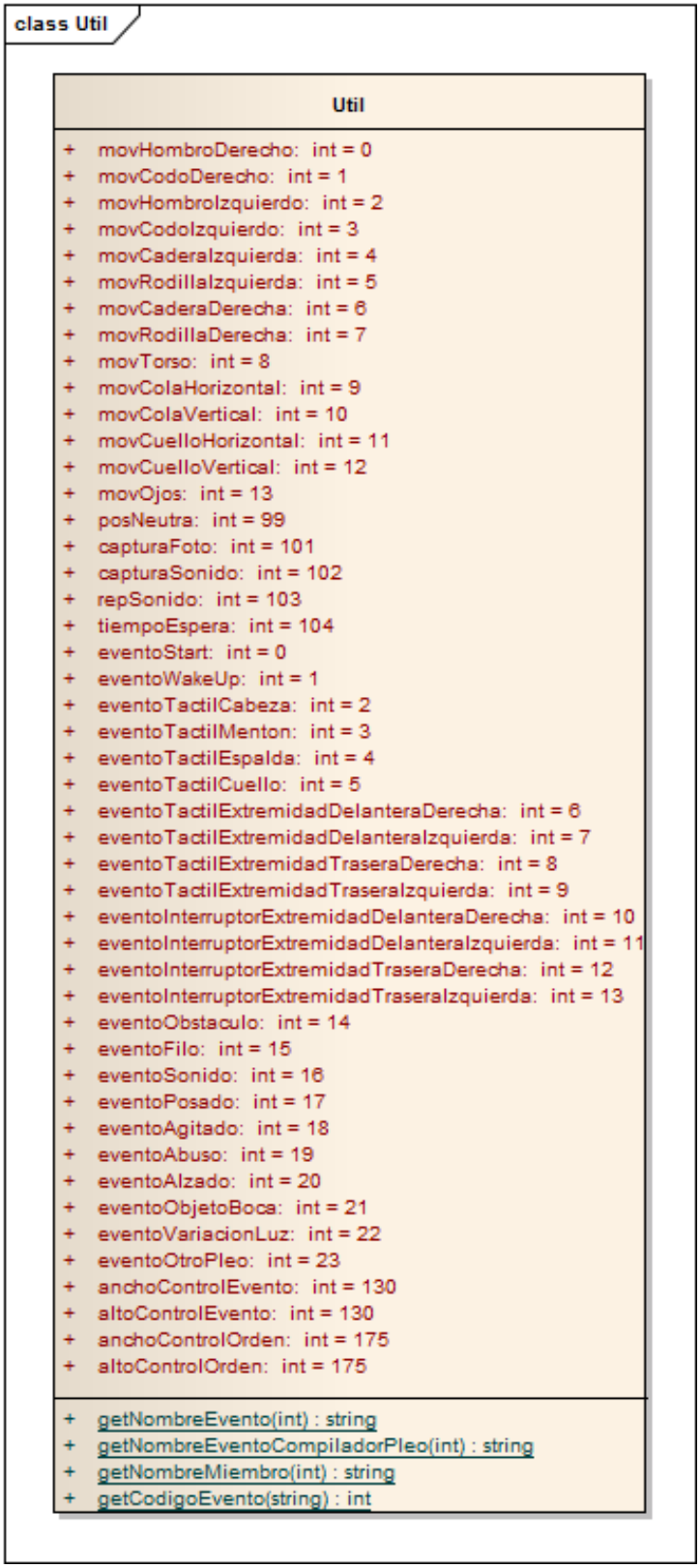


Figura 6.7 Diagrama de la clase Util

6.3 Diagramas de Secuencia

6.3.1 Insertar Evento

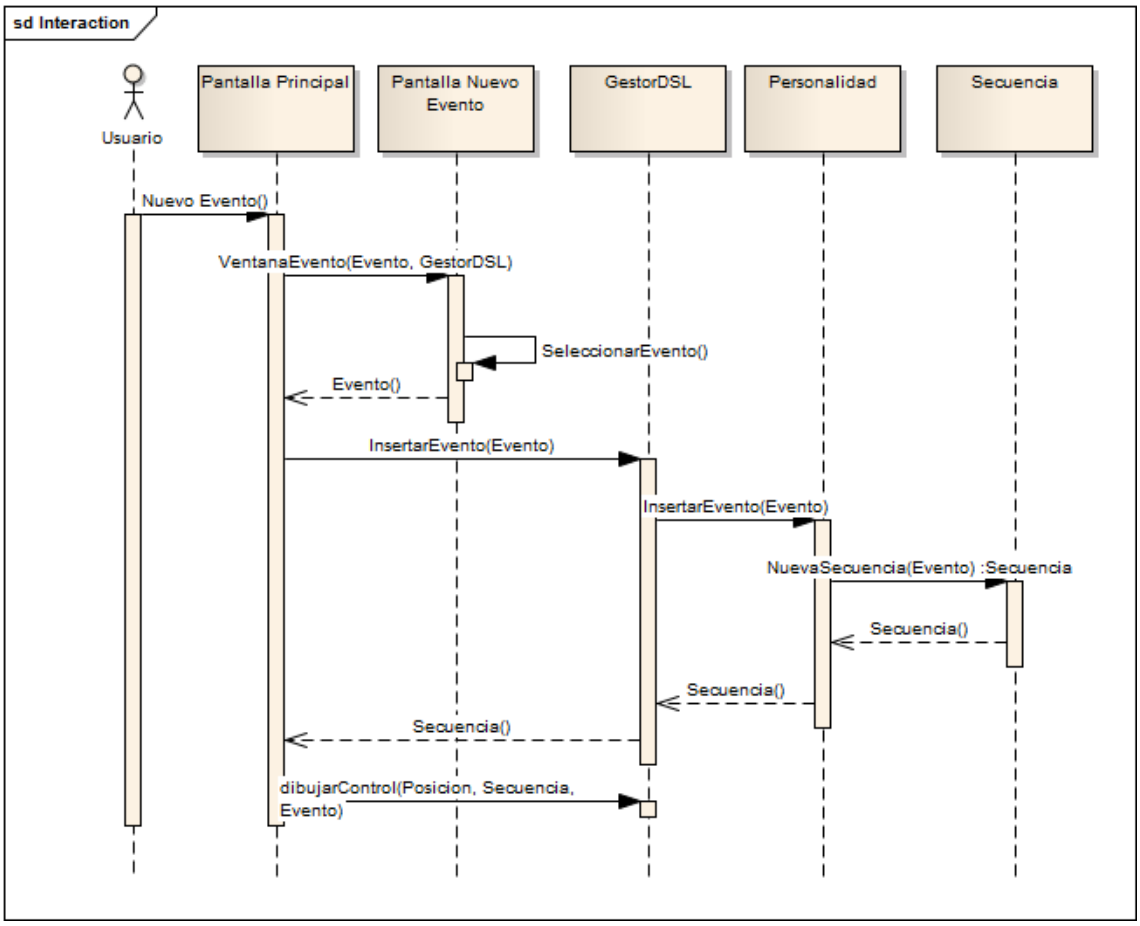


Figura 6.8 Diagrama de secuencia para la operación insertar un evento

El anterior diagrama muestra la secuencia de la acción de añadir un nuevo evento. El usuario elige añadir un nuevo evento en la pantalla principal, esta realiza una llamada a una nueva ventana para mostrar los eventos disponibles, verifica que no esté incluido ya y devuelve el nuevo evento. Una vez se recibe el nuevo evento se llama al GestorDSL para que inserte el nuevo evento en la personalidad, una vez insertado devuelve la secuencia correspondiente. Por último la pantalla principal solicita a GestorDSL que muestre en el canvas de la pantalla principal el nuevo evento.

6.3.2 Eliminar Evento

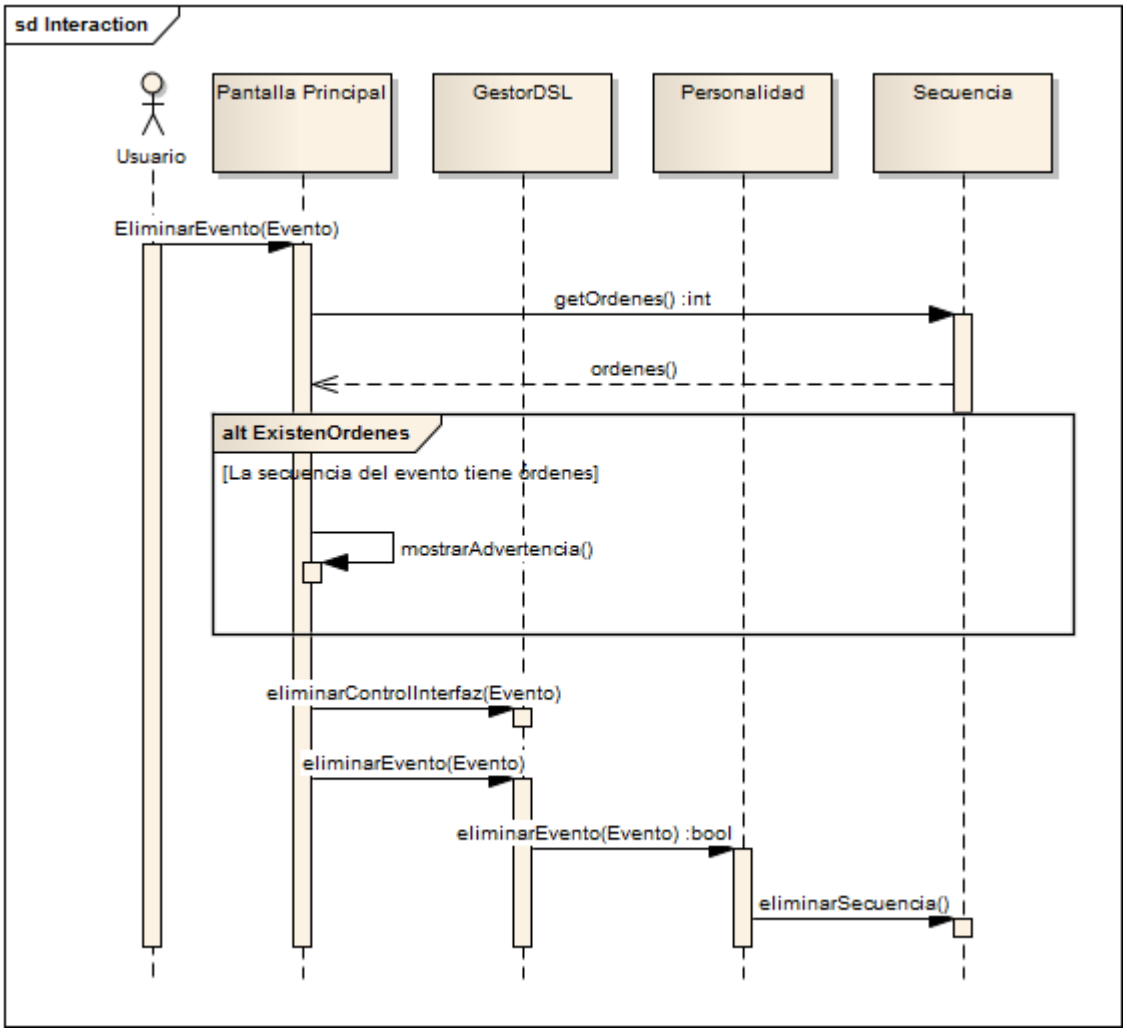


Figura 6.9 Diagrama de secuencia para la operación eliminar un evento

El diagrama para eliminar un evento muestra las siguientes acciones. Primero se solicita por parte del usuario eliminar un evento, se comprueba el número de órdenes que tiene dicho evento y si este las tiene se muestra un mensaje de advertencia indicando que serán borradas de continuar con la operación. A continuación se elimina en control de la interfaz de usuario y posteriormente de la parte lógica haciendo las correspondientes llamadas al GestorDSL. Éste a su vez llama a eliminarEvento de la personalidad actual, pasando el evento a eliminar, si se produce sin ningún tipo de fallo devuelve el valor true. La personalidad llama a la secuencia que se corresponde con el evento que se desea eliminar y una vez que se encuentra se borra totalmente.

6.3.3 Insertar Orden

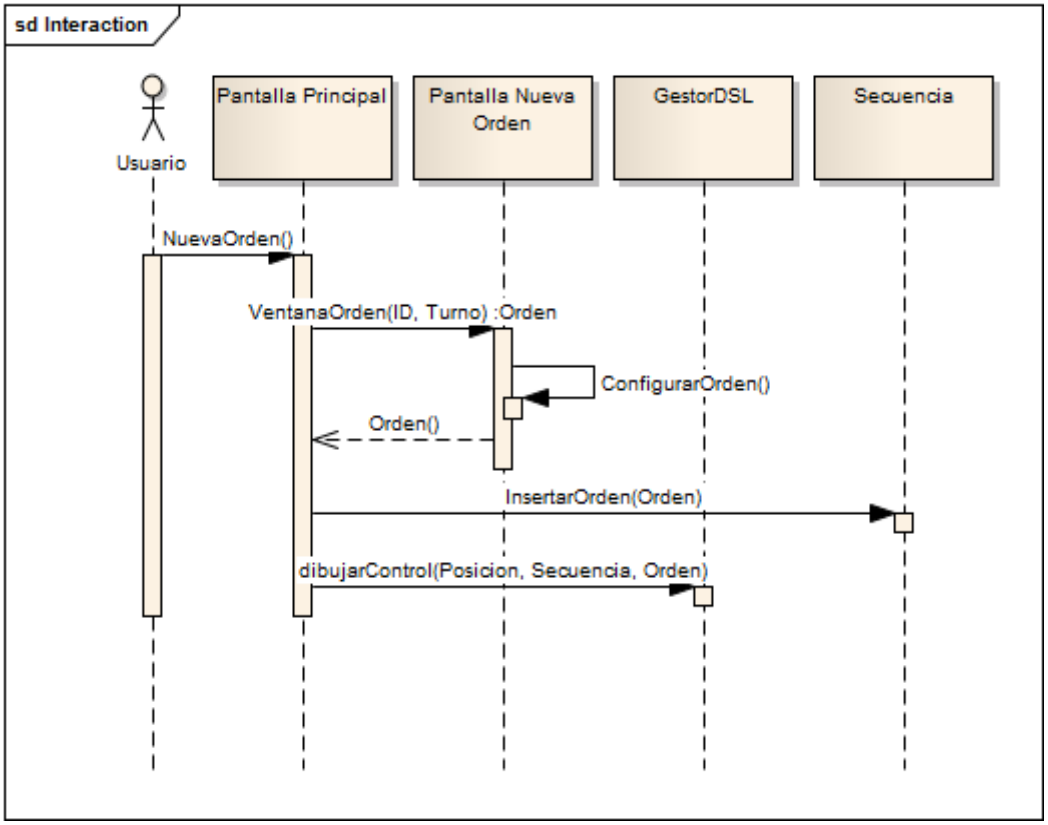


Figura 6.10 Diagrama de secuencia para la operación insertar una orden

Cuando se desee insertar una orden, y en función de la secuencia activa, el usuario tendrá activado el botón de la orden que desee insertar. La pantalla principal mostrará una nueva pantalla donde configurar la orden. Cuando se termine de configurar se devuelve la orden y se llama al método insertar de la secuencia activa, pasando la orden creada. Por último se dibuja la orden como un control en la pantalla principal.

Para el caso de **editar orden** el procedimiento es el mismo, llamando a editar orden de secuencia en vez insertar orden y obviando el dibujado del control pues ya se encuentra en la interfaz.

6.3.4 Eliminar Orden

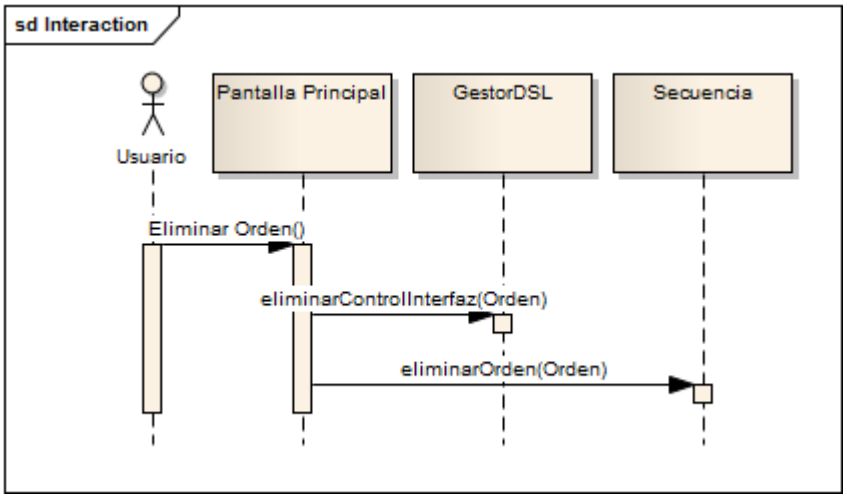


Figura 6.11 Diagrama de secuencia para la operación eliminar una orden

El usuario solicita la eliminación de una orden, esta se elimina de la interfaz de usuario y posteriormente de la secuencia a la que pertenece.

6.3.5 Aumentar Turno Orden

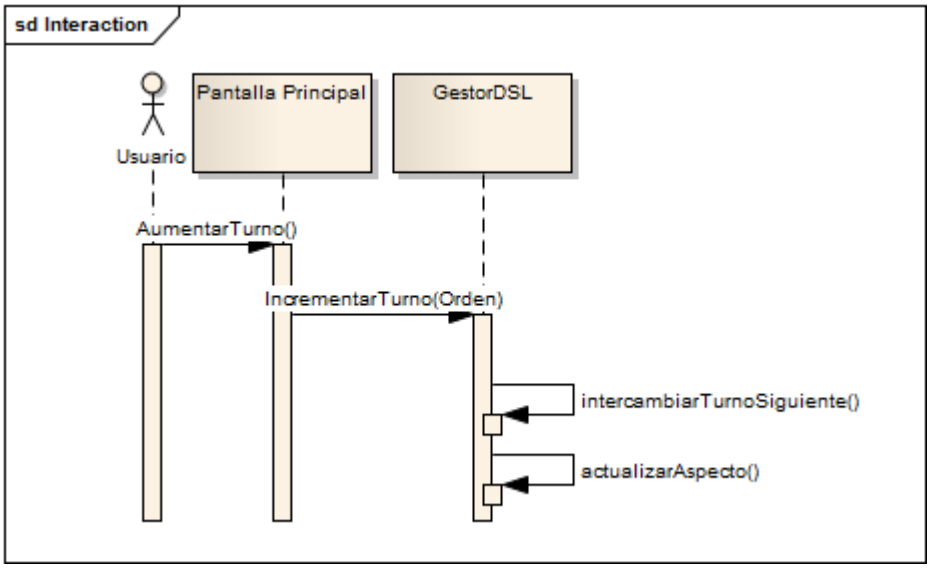


Figura 6.12 Diagrama de secuencia para la operación aumentar turno de orden

El usuario selecciona una orden y ejecuta la acción de aumentar turno, o retrasar su ejecución dentro de la secuencia, se llama al método incrementarTurno y se le pasa la Orden que se desea editar. El GestorDSL intercambia el turno de la orden que se le pasa con la siguiente y se actualiza el aspecto del control que la contiene.

Para el caso de **disminuir turno orden** es igual, llamando a decrementarTurno e intercambiando el turno con la orden anterior.

6.3.6 Compilar

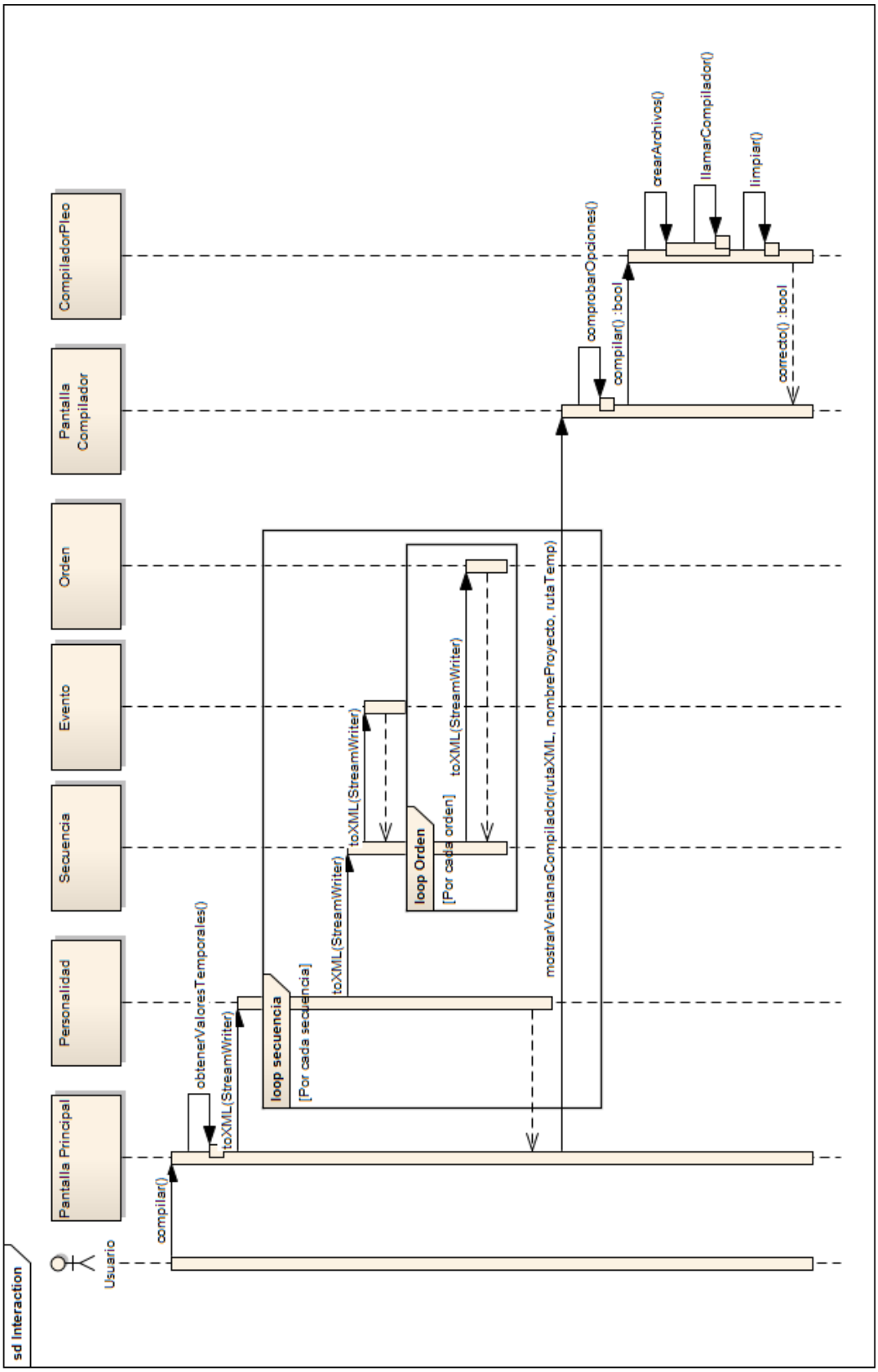


Figura 6.13 Diagrama de secuencia para la operación compilar

La operación de compilado se requiere por el usuario cuando la personalidad se encuentre finalizada. Para cada secuencia, y para cada orden de cada secuencia, se escribe en un fichero XML los datos correspondientes. Para ello se llama al método toXML() de cada uno de estos objetos pasando un StreamWriter previamente creado, en este StreamWriter se irán escribiendo todas las ordenes y eventos que constituyen la personalidad. Este es el momento en el que se transforma el diagrama DSL en un fichero XML.

Posteriormente se muestra una pantalla en la que se pueden introducir opciones adicionales para el compilado. Estas opciones son la ruta de destino, la forma de vida artificial para la que se está compilando, la versión de dicha forma de vida y la ruta del compilador correspondiente a la forma de vida.

Al compilar se usa el fichero XML generado anteriormente para crear los archivos necesarios para llamar al compilador de la forma de vida escogida, en este diagrama Pleo, se llama al compilador y se limpian los archivos temporales.

6.4 Diagramas de Actividades

6.4.1 Compilar Personalidad

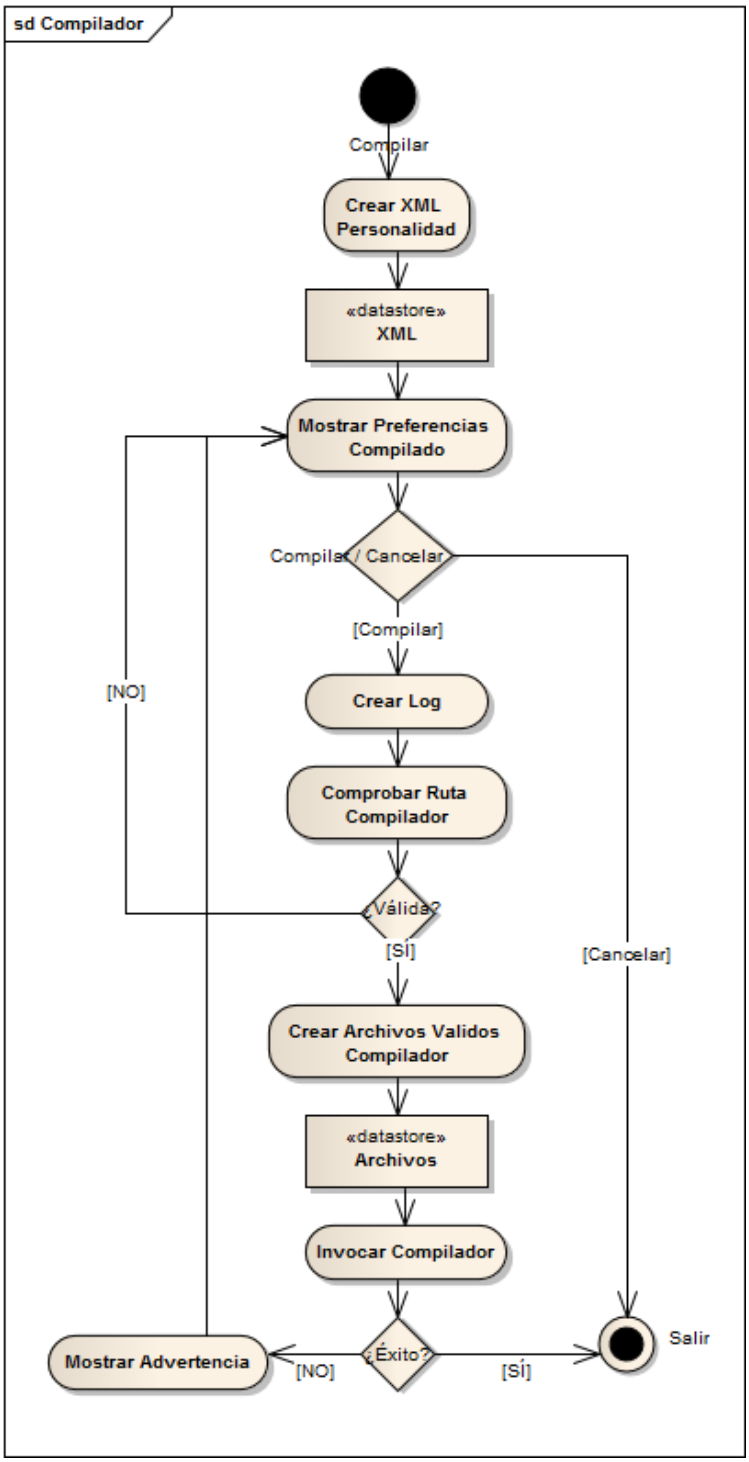


Figura 6.14 Diagrama de actividad compilar

El anterior diagrama representa la actividad compilar, se ha querido crear para obtener mayor detalle de esta operación.

6.5 Diseño de los Ficheros Utilizados

A continuación se presenta información acerca de la estructura de los ficheros que guarda y utiliza la aplicación.

6.5.1 Proyecto

Para poder guardar, y posteriormente, cargar los proyectos que el usuario vaya realizando, la personalidad se almacenará en un fichero binario donde el usuario indique.

Este fichero binario creado por la aplicación en el momento que el usuario desee guardar es un fichero que almacena la clase Personalidad, para ello dicha clase y todas las que utiliza (Secuencia, Orden, Evento) han sido marcadas como **serializable**.

6.5.2 XML Personalidad

Para dotar de independencia el compilador del resto de la aplicación para que, si es necesario, sea fácil modificarlo para otras versiones de la forma de vida artificial elegida, la comunicación entre el Gestor de la Personalidad y este se realiza con el intercambio de un fichero que almacena toda la información de la personalidad creada por el usuario.

Este fichero se almacena de forma temporal en una carpeta que el propio sistema adjudica (normalmente la carpeta temporal de las aplicaciones). Es en este directorio donde la personalidad guarda el fichero XML y donde el compilador accede a este para convertirlo al formato adecuado para la forma de vida artificial. Este fichero se borra una vez la compilación ha finalizado.

La estructura interna de este fichero s puede observar con el siguiente ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<Personalidad>
	<Secuencia>
		<Evento>0</Evento>
		<Ordenes>
			<Orden>
				<Tipo>Neutro</Tipo>
				<Id>0</Id>
				<Turno>1</Turno>
			</Orden>
		</Ordenes>
	</Secuencia>
	<Secuencia>
		<Evento>2</Evento>
		<Ordenes>
			<Orden>
				<Tipo>Captura Sonido</Tipo>
				<Id>2</Id>
				<Turno>1</Turno>
				<Nombre>Grabacion1</Nombre>
				<Tiempo>10</Tiempo>
			</Orden>
			<Orden>
				<Id>3</Id>
				<Turno>2</Turno>
				<Tipo>Movimiento Repetitivo</Tipo>
				<Miembro>13</Miembro>
				<AnguloFin>50</AnguloFin>
```

```
                            <AnguloInicio>0</AnguloInicio>
                            <Repeticiones>5</Repeticiones>
                        </Orden>
                </Ordenes>
        </Secuencia>
</Personalidad>
```

En el anterior XML, de ejemplo, se han definido dos eventos que disparan dos órdenes cada uno. Los eventos están identificados por enteros que se corresponden con diferentes sucesos externos.

Cada evento tiene anidadas tantas órdenes como se hallan añadido. Dependiendo del tipo de la orden tendrán más o menos campos. Para todas las órdenes son comunes los campos ID, que es un identificador único para cada orden y Turno, que es un entero que informa del turno dentro de la secuencia de esa orden.

6.5.3 LOG

El archivo de log se crea en la compilación para recoger los posibles errores y fallos que se produzcan en el transcurso de esta.

Es un simple fichero de texto (.txt) que se almacena en la carpeta temporal por defecto. Es posible indicar en las preferencias una ruta donde guardar el último Log (se sobrescribe si hay un archivo con el mismo nombre). Si no se ha especificado una ruta, el archivo será borrado una vez se haya llevado a cabo la compilación.

También la herramienta facilita guardar este log cuando se produzca algún fallo, para ello muestra al usuario una ventana donde puede indicar la ruta donde desea guardarlo.

6.5.4 Archivos del compilador Pleo

Cada compilador implementado tiene sus propios archivos de entrada. En este caso Pleo tiene como entrada un archivo .p, un archivo .upf, y si el usuario desea reproducir alguna animación o sonido, los directorios que los contengan.

Este directorio de archivos y carpetas se almacena en la misma carpeta temporal que los archivos anteriores, y una vez la compilación se ha llevado a cabo son eliminados.

El formato interno de estos archivos está detallado en la parte de aspectos teóricos, dentro de “PDK”.

El archivo resultante de la compilación, un archivo con extensión .urf, se guardará en la ubicación que el usuario haya elegido en la ventana de compilación. Es posible almacenar una ruta por defecto donde ubicar los archivos resultantes dentro de la opción de preferencias en la ventana principal. El archivo .urf debe ser copiado en la raíz de la tarjeta SD que se vaya a introducir a Pleo.

6.5.5 Batch

De forma temporal se crea un pequeño archivo Batch para ser ejecutado y que es el que inicia el compilado pasando los argumentos correspondientes. Este fichero se almacena en la carpeta temporal. Tiene por objetivo establecer el Path correcto y llamar al compilador, además aquí se especifica que la salida se envíe a un fichero que posteriormente se incluirá en el log.

El aspecto de este fichero Batch suele ser algo parecido a lo siguiente:

```
PATH %PATH%D:\Documentos de usuarios\Usuario\Mis documentos\PleoDevelopmentKit\bin;
C:
cd C:\Users\Usuario\AppData\Local\Temp\PLEO_152458_2962011\
Ugobe_project_tool Prueba.upf rebuild >
C:\Users\Usuario\AppData\Local\Temp\PLEO_152458_2962011\resultadoCompilacion.txt
```

Este Batch es generado en el momento conforme a ciertos valores en tiempo de ejecución. La primera línea establece el Path a la ruta del compilador, en este caso para Pleo, para poder utilizar el compilador de dicha forma de vida. La segunda y tercera línea establece el directorio actual, que es el lugar donde se han generado los archivos de entrada del compilador. La cuarta línea (que se encuentra partida por ser demasiado larga) llama al compilador pasándole el archivo .upf creado y direcciona la salida a un archivo de texto.

6.6 Diseño de la Interfaz

A continuación se mostrarán las ventanas que se han desarrollado para la herramienta y su aspecto definitivo.

Las ventanas y diálogos mostrados tienen establecido botón por defecto de aceptación y cancelación y el foco siempre se encontrará en el primer recuadro habilitado.

6.6.1 Cuadro de inicio

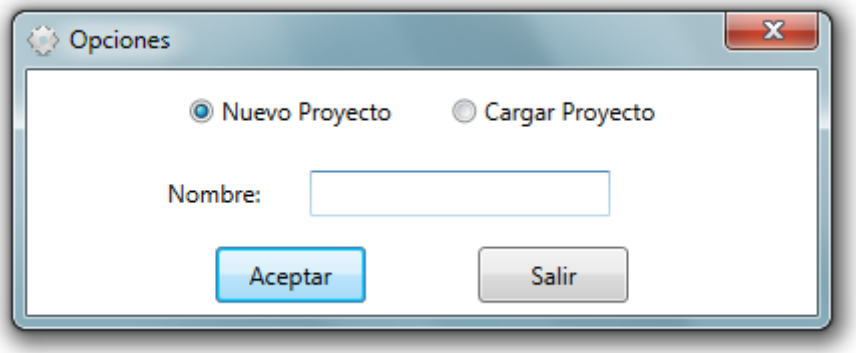


Figura 6.15 Cuadro de inicio nuevo proyecto

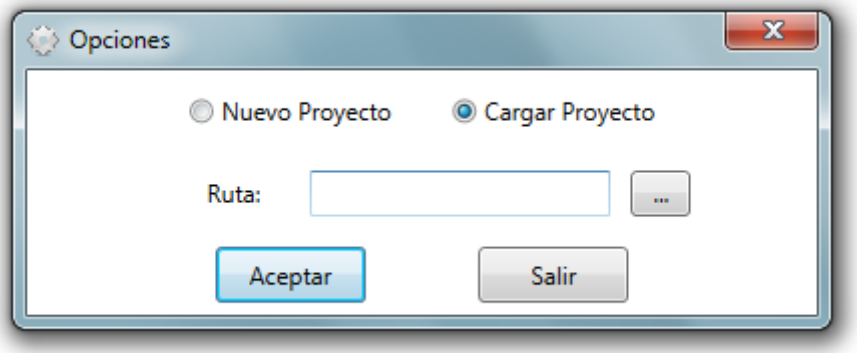


Figura 6.16 Cuadro de inicio cargar proyecto

Al iniciar la herramienta el primer diálogo mostrado será el anterior cuadro. En dicho cuadro es posible elegir si crear un nuevo proyecto, en cuyo caso se escribirá el nombre de dicho proyecto o si prefiere cargar un nuevo proyecto, y el usuario tendrá que especificar la ruta de dicho archivo de proyecto.

Este cuadro está compuesto por dos radio button que modifican el aspecto del cuadro según se activen o desactiven. Debajo de estos se encuentran, visibles o no, un cuadro de texto y tres botones.

6.6.2 Pantalla Principal

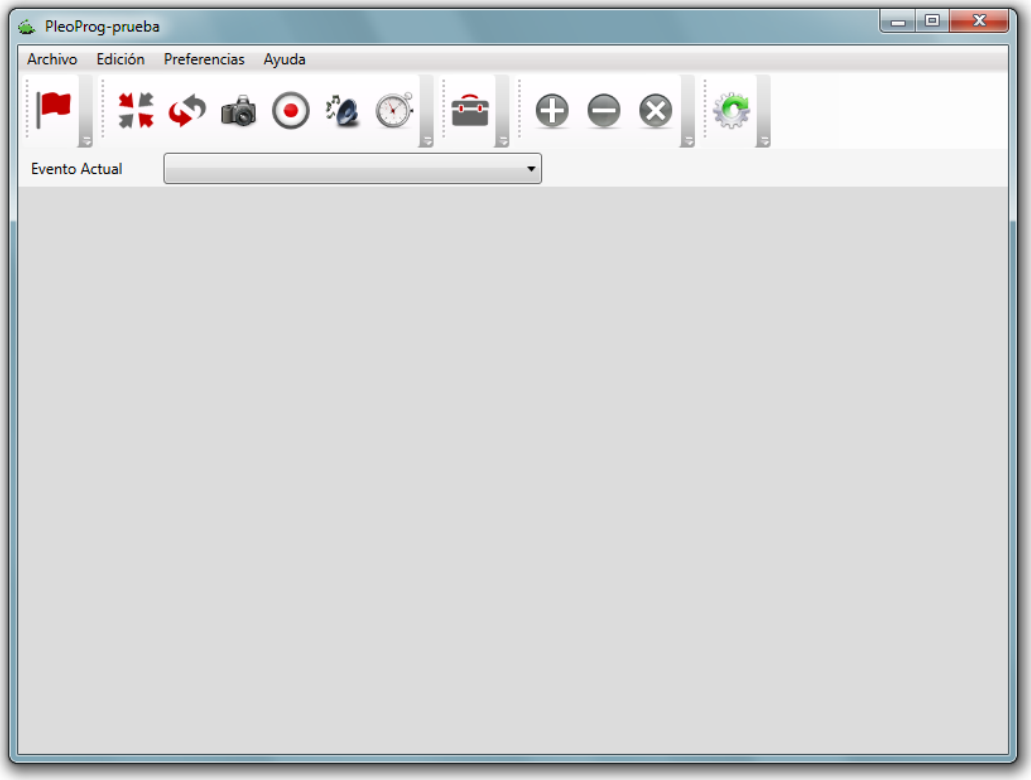


Figura 6.17 Pantalla principal de la aplicación

La figura anterior muestra el aspecto de la ventana principal definitiva. De arriba abajo se observan dos áreas, la primera compuesta por tres barras de menús y herramientas y la segunda que es el entorno de trabajo donde se crearán los diagramas.

La primera área está dividida en tres barras. La primera de ellas es la **barra de menú** donde se encuentran las opciones de Archivo (nuevo, guardar, cargar, cerrar, salir), Edición (aumentar turno, disminuir turno y eliminar), Preferencias (para guardar las preferencias del usuario en cuanto a las rutas del compilador, ruta de destino del archivo compilado y ruta del log) y por último la opción Ayuda (ver ayuda, página de PleoProg, acerca de PleoProg). La segunda barra es la **barra de herramientas** en la que se encuentran las opciones de añadir evento, añadir cada una de las órdenes (posición neutra, movimiento, captura de imagen, captura de sonido, reproducción de sonido y tiempo de espera), añadir animación, aumentar turno, disminuir turno, eliminar y por último compilar. Todos los botones de evento y órdenes son interruptores que mientras permanezcan pulsados se podrán añadir el evento u orden sin tener que volver a pulsarlo. La tercera barra es una **barra de estado** donde se indica el evento actual y en el que se añadirá la próxima orden que se inserte.

La segunda zona es el **área de trabajo** y es donde el usuario colocará cada uno de los eventos y órdenes, solo debe activar el botón que represente lo que desea añadir y hacer clic en la posición del área de trabajo donde desea añadirlo.

Este aspecto final solo difiere del diseño previo en la colocación de la barra de estado.

6.6.3 Cuadro de eventos

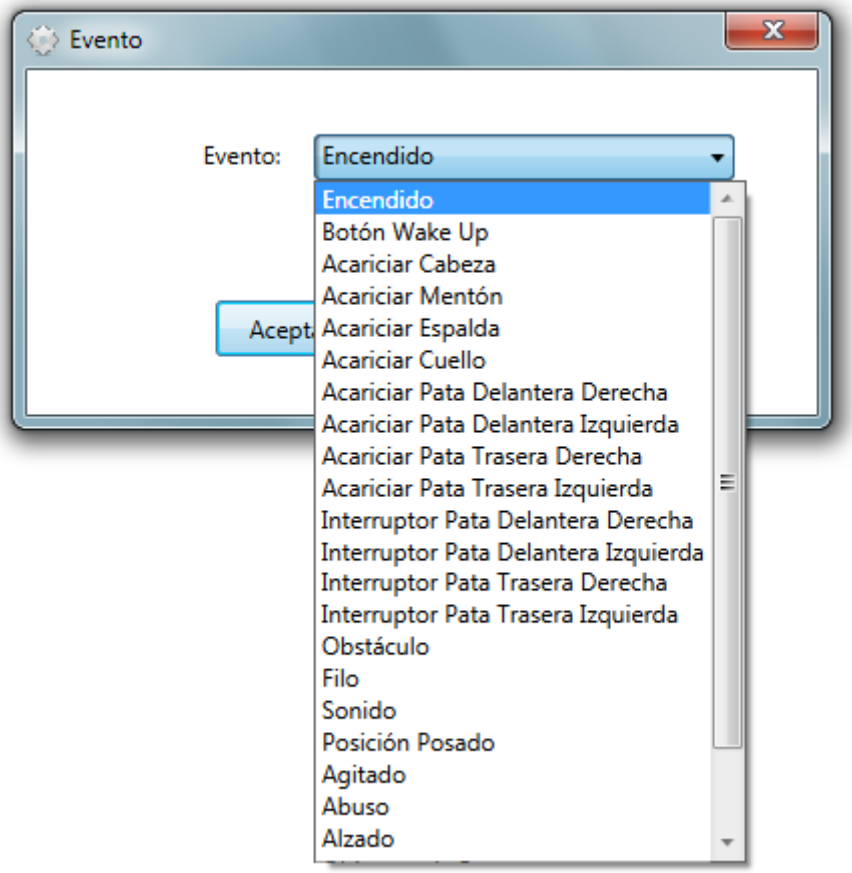


Figura 6.18 Cuadro añadir evento

El anterior cuadro se muestra al usuario cuando el botón activo es el de añadir un evento y hace clic en el área de trabajo. Se muestra un cuadro como el de la figura donde es posible escoger entre los eventos disponibles el que desee añadir.

Dicho cuadro consta de un combo box y dos botones.

6.6.4 Órdenes

Para cada una de las órdenes que el usuario desee añadir al área de trabajo existen diferentes cuadros para introducir información relativa a estas órdenes. A continuación cada una de las órdenes, según se encuentran situadas en la interfaz de izquierda a derecha, y el cuadro correspondiente para configurarlas.

Para la orden de **posición neutra** no se muestra ningún cuadro puesto que no es necesario ningún dato extra.

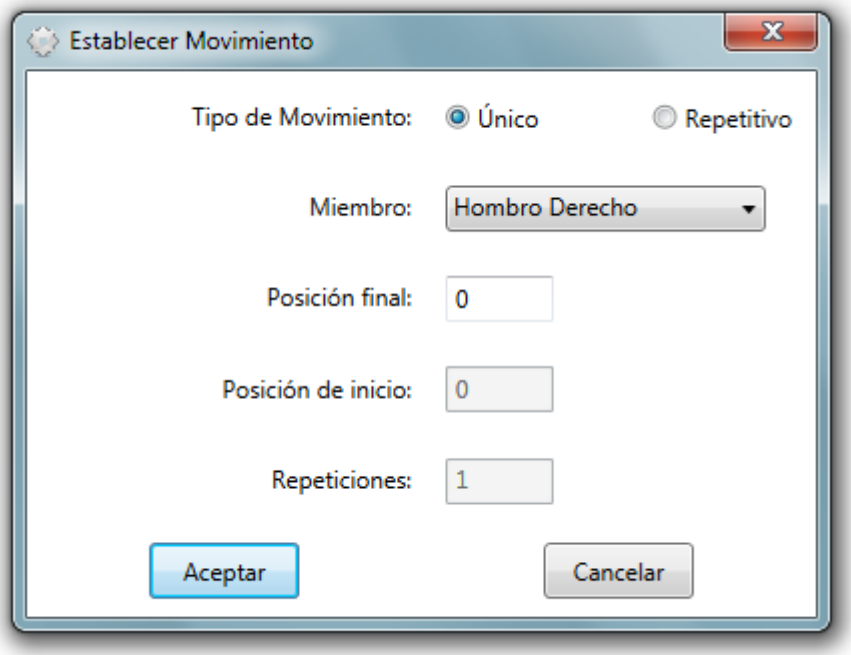


Figura 6.19 Cuadro inserción orden de movimiento

El anterior cuadro se muestra cuando el usuario desea añadir una orden de **movimiento**. En la parte superior hay dos radio button que activan o desactivan los campos necesarios según el tipo de movimiento. Tanto para los movimientos únicos como para los repetitivos se ha de elegir el miembro a mover en el combo box y la posición final de movimiento.

Si se trata de un movimiento repetitivo será necesario indicar la posición de inicio del movimiento y el número de repeticiones en los respectivos text box.

Al aceptar se validan todos los datos y se añade la orden si es coherente.

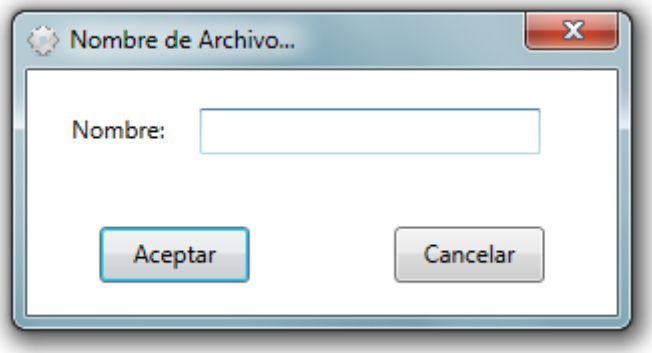


Figura 6.20 Cuadro de inserción captura imagen

Cuando se desee añadir la orden de **captura de imagen** se mostrará un cuadro como en anterior donde el usuario debe introducir en el text box el nombre que tendrá la imagen que se guardará.

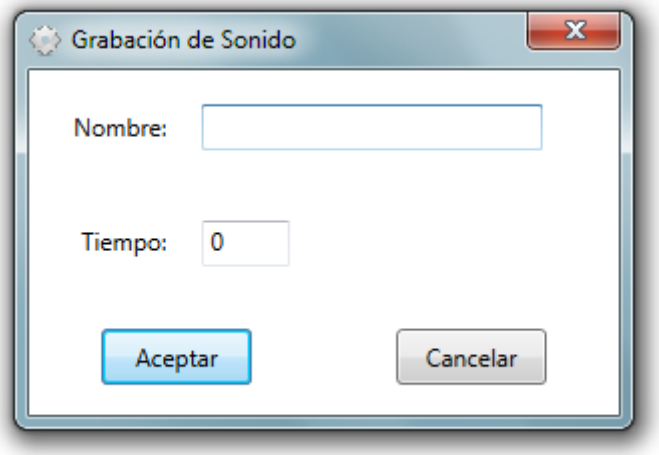


Figura 6.21 Cuadro de captura sonido

El anterior cuadro se mostrará para configurar la orden de **captura de sonido**. Dos text box donde el usuario indicará el nombre que desea dar a dicha grabación y el tiempo de grabación.

Para la orden de **reproducción de sonido** se abrirá una ventana normal para seleccionar el archivo.

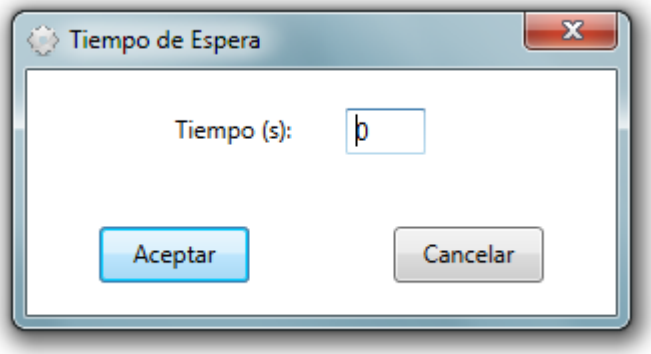


Figura 6.22 Cuadro de tiempo en espera

El anterior cuadro se muestra cuando se desea añadir una orden de **tiempo en espera**. Consta simplemente de un text box donde introducir los segundos que la forma de vida se mantendrá en espera.

Para la orden de inserción de **animación** se muestra una ventana para seleccionar el archivo, como en el caso de reproducción de sonido.

6.6.5 Preferencias

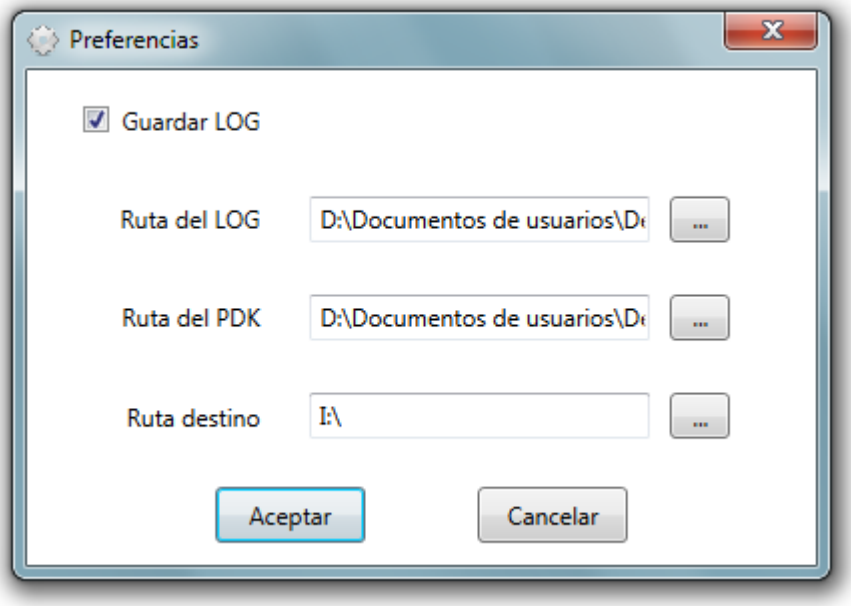


Figura 6.23 Cuadro de preferencias

Este cuadro se muestra cuando se presiona sobre preferencias en la barra de menú. Hay un check box para indicar si se desea guardar el log y tres text box no editables pero que muestran la ruta seleccionada en cada una de las ventanas de selección de directorio que lanzan los botones situados a la derecha de dichos text box.

6.6.6 Compilador

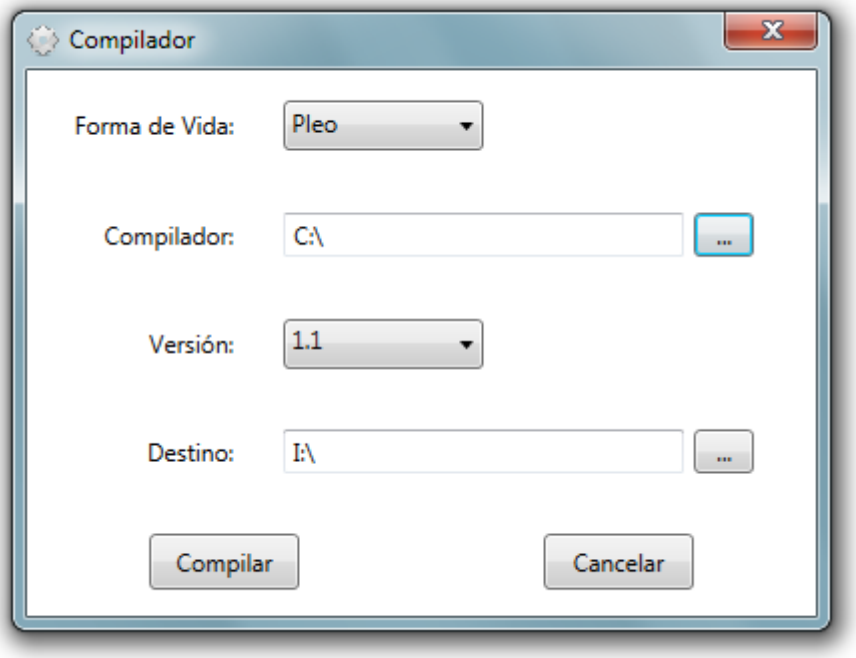


Figura 6.24 Cuadro compilador

El cuadro compilador muestra un combo box para seleccionar la forma de vida artificial para la que se va a compilar, actualmente únicamente Pleo, un text box y un button para seleccionar la ruta del compilador de la forma de vida, un check box para seleccionar la versión de la forma de vida y un text box y un button para seleccionar la carpeta destino del archivo resultante.

6.6.7 Ayuda

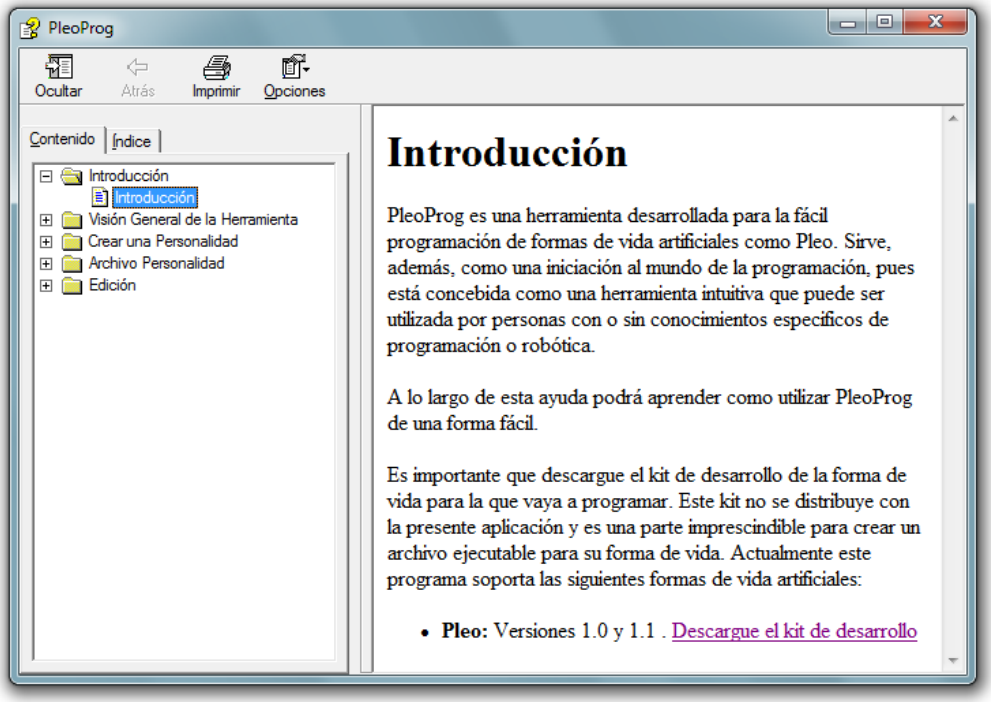


Figura 6.25 Ayuda

El cuadro de ayuda se muestra cuando el usuario presiona F1 o cuando hace clic sobre el menú de ayuda.

Se trata de una ayuda breve creada para servir de apoyo y base para las funciones más básicas.

6.6.8 Iconos

Los iconos que representan nuevo evento, posición neutra, movimiento y animación son de freeiconsweb.com (http://www.freeiconsweb.com/), son parte de la serie office icons y se encuentran bajo una licencia "Free for commercial use (Include link to authors website)".

Los iconos que representan la captura de imagen y reproducción de sonido son parte del Gnome Project (http://art.gnome.org/themes/icon) y se encuentran bajo licencia GPL (http://www.gnu.org/copyleft/gpl.html)

Los iconos de grabación de sonido, tiempo en espera, aumentar turno, disminuir turno y eliminar son de Oxygen (http://www.oxygen-icons.org) y se encuentra bajo una licencia GPL (http://www.gnu.org/copyleft/gpl.html)

6.7 Especificación Técnica del Plan de Pruebas

A continuación se describen las pruebas para llevar a cabo. Estas pruebas se llevarán a cabo después de la finalización de cada una de las funcionalidades que posibilitarán realizar las pruebas.

La herramienta se ha probado bajo un equipo con la siguiente configuración:

- Windows 7 64 bits
- Microsoft Visual Studio 2010
- Intel Core Quad Q6600. 6GB RAM.

6.7.1 Pruebas Unitarias

A continuación se muestran las pruebas unitarias que se comenzaron a desarrollar en la fase de análisis. Estas pruebas se llevarán a cabo cuando cada una de las clases que interviene en cada caso de uso haya sido finalizada. En caso de detectar algún fallo se toma nota y se modifica al finalizar las pruebas de mismo caso de uso.

Se presentan divididas en casos de uso ya vistos anteriormente.

Caso de Uso 1: Insertar Evento	
Prueba	**Resultado Esperado**
Guardar un evento cualquiera. Clases que intervienen: Personalidad, Secuencia y Evento y Util.	Se guarda correctamente un evento nuevo. La clase Personalidad posee una secuencia más.
Prueba	**Resultado Esperado**
Cancelar la Operación	No se guarda ningún evento.

Caso de Uso 2:Eliminar Evento	
Prueba	**Resultado Esperado**
Eliminar un evento existente. Clases que intervienen: Personalidad, Secuencia y Evento y Util.	Se elimina correctamente un evento. La clase Personalidad posee una secuencia menos.
Prueba	**Resultado Esperado**
Cancelar la Operación	No se elimina ningún evento.

Caso de Uso 3: Insertar Orden	
Prueba	**Resultado Esperado**
Guardar una orden cualquiera. Clases que intervienen: Secuencia, Orden y Util.	Se guarda correctamente una orden nueva. La clase Secuencia tiene una orden más.

Prueba	**Resultado Esperado**
Cancelar la Operación	No se guarda ninguna orden.

Caso de Uso 4: Modificar Orden	
Prueba	**Resultado Esperado**
Se modifica una orden existente. Clases que intervienen: Secuencia, Orden y Util.	Se modifican los datos de una orden. La clase Secuencia posee la misma orden pero modificada.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 5: Eliminar Orden	
Prueba	**Resultado Esperado**
Eliminar una orden existente. Clases que intervienen: Secuencia, Orden y Util.	Se elimina una orden. La clase Secuencia posee una orden menos.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 6: Aumentar Turno	
Prueba	**Resultado Esperado**
Aumentar turno de una orden. Clases que intervienen: Secuencia, Orden y Util.	La orden tiene un turno mayor. La clase Secuencia modifica dos órdenes, la orden que modifica su turno y la orden siguiente en la secuencia que también modifica su turno.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 7: Disminuir Turno	
Prueba	**Resultado Esperado**
Disminuir turno de una orden Clases que intervienen: Secuencia, Orden y Util.	La orden tiene un turno menor. La clase Secuencia modifica dos órdenes, la orden que modifica su turno y la orden anterior en la secuencia que también modifica su turno.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

6.7.2 Pruebas de Integración y del Sistema

A continuación se detallan las pruebas de integración y del sistema. Están divididas en casos de uso.

Dichas pruebas se llevarán a cabo cuando se posea una versión inicial de la aplicación y se irá tomando nota de los fallos encontrados para ser corregidos al finalizar las pruebas de cada caso de uso. Estas pruebas son más complejas que las unitarias puesto que son varias las partes de la herramienta las que intervienen y deben funcionar correctamente.

Caso de Uso 1: Insertar Evento	
Prueba	**Resultado Esperado**
Insertar un evento no existente. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Evento y Util.	La clase Personalidad posee una nueva instancia de la clase Secuencia más. La clase Secuencia almacena correctamente los datos del evento introducido. La clase GestorDSL crea y dibuja el control apropiado en el área de trabajo.
Prueba	**Resultado Esperado**
Insertar un evento repetido. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Evento y Util.	El sistema detecta que dicho evento ya existe dentro de Personalidad y notifica dicha incidencia al usuario para que elija otro Evento.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 2:Eliminar Evento	
Prueba	**Resultado Esperado**
Eliminar un evento sin órdenes. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Evento y Util.	La clase Personalidad posee una Secuencia menos. GestorDSL busca y elimina de la interfaz el control que representa dicho evento.
Prueba	**Resultado Esperado**
Eliminar un evento con órdenes. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Evento y Util.	El sistema detecta que se intenta borrar un evento que posee órdenes. Notifica al usuario que si continua borrará también dichas órdenes. En caso afirmativo, Personalidad posee una Secuencia menos y GestorDSL busca y elimina de la interfaz el control que representa dicho evento y las órdenes que posee.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 3: Insertar Orden	
Prueba	**Resultado Esperado**
Insertar una orden existiendo un evento. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	La clase Personalidad asigna a Secuencia dicha Orden y Secuencia posee una Orden más. GestorDSL crea y dibuja el control apropiado en el área de trabajo.
Prueba	**Resultado Esperado**
Insertar una orden sin que exista ningún evento o que no se encuentre ninguno activo. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Evento, Orden y Util.	El sistema detecta que no existe ningún evento o que ninguno se encuentra activo. Notifica al usuario que añada o seleccione algún evento de los disponibles y lo vuelva a intentar. No se añade ninguna Orden a ninguna Secuencia.
Prueba	**Resultado Esperado**
Insertar una orden con	El sistema detecta que las opciones introducidas no son

opciones incorrectas. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	correctas. Notifica al usuario que revise las opciones introducidas. No se añade ninguna Orden a ninguna Secuencia.
Prueba	**Resultado Esperado**
Insertar una orden que incluya un archivo de recurso (sonido o animación) superior a 1 MB. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	El sistema detecta que se intenta adjuntar un archivo con un tamaño superior a 1MB. Notifica al usuario que el archivo es demasiado grande. No se añade ninguna Orden a ninguna Secuencia.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 4: Modificar Orden	
Prueba	**Resultado Esperado**
Modificar una orden con opciones correctas. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	Personalidad realiza los cambios en la Orden de la Secuencia y GestorDSL realiza los cambios gráficamente del control que representa dicha Orden en el área de trabajo.
Prueba	**Resultado Esperado**
Modificar una orden con opciones incorrectas Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	El sistema detecta que las opciones de la Orden no son correctas. Notifica al usuario para que lo vuelva a intentar o cancele dichos cambios. La Orden de la Secuencia no se modifica.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 5: Eliminar Orden	
Prueba	**Resultado Esperado**
Eliminar una orden existente. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	La Personalidad solicita a Secuencia que se elimine una Orden. Secuencia posee una Orden menos. GestorDSL elimina el control pertinente del área de trabajo.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 6: Aumentar Turno	
Prueba	**Resultado Esperado**
Se aumenta el turno de una orden que no esté en último lugar. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	La clase Personalidad solicita a Secuencia cambiar el turno de la Orden seleccionada y de la Orden siguiente intercambiando los turnos. GestorDSL intercambia el lugar y el turno de la Orden seleccionada y de la Orden siguiente.
Prueba	**Resultado Esperado**

Se aumenta el turno de una orden que se encuentra en último lugar. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	El sistema detecta que la Orden seleccionada se encuentra en último lugar dentro de la Secuencia y no posee órdenes posteriores por lo que no se realiza ningún tipo de cambio.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 7: Disminuir Turno	
Prueba	**Resultado Esperado**
Se disminuye el turno de una orden que no esté en primer lugar. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	La clase Personalidad solicita a Secuencia cambiar el turno de la Orden seleccionada y de la Orden anterior intercambiando los turnos. GestorDSL intercambia el lugar y el turno de la Orden seleccionada y de la Orden anterior.
Prueba	**Resultado Esperado**
Se disminuye el turno de una orden que se encuentra en primer lugar. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	El sistema detecta que la Orden seleccionada se encuentra en primer lugar dentro de la Secuencia y no posee órdenes anteriores por lo que no se realiza ningún tipo de cambio.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 8: Compilar	
Prueba	**Resultado Esperado**
Compilar un programa válido. Clases que intervienen: Personalidad, Secuencia, Orden, Evento, Compilador, CompiladorPleo y Util.	Personalidad guarda en un fichero XML todas sus Secuencias con Eventos y Órdenes y pasa este fichero a Compilador. En este caso, CompiladorPleo realiza una traducción de dicho fichero a código de entrada para el compilador de Pleo y realiza la llamada a dicho compilador. En la ruta destino se crea un archivo ejecutable por Pleo.
Prueba	**Resultado Esperado**
Compilar un programa sin eventos. Clases que intervienen: Personalidad, Secuencia, Orden, Evento, Compilador, CompiladorPleo y Util.	El sistema detecta que la Personalidad no tiene ninguna Secuencia. Notifica al usuario que no se puede compilar un programa vacio. No se realiza la compilación.
Prueba	**Resultado Esperado**
Compilar un programa cuando la ruta del compilador es errónea. Clases que intervienen: Personalidad, Secuencia, Orden, Evento, Compilador,	El sistema detecta que la carpeta del compilador especificada no es correcta. Notifica al usuario y le sugiere cambiar dicha ruta. No se realiza ninguna compilación.

CompiladorPleo y Util.	
Prueba	**Resultado Esperado**
Compilar un programa cuando la ruta de destino es errónea. Clases que intervienen: Personalidad, Secuencia, Orden, Evento, Compilador, CompiladorPleo y Util.	El sistema detecta que la ruta de destino es errónea. Notifica al usuario para que cambie dicha ruta. No se realiza ninguna compilación.
Prueba	**Resultado Esperado**
Compilar un programa no válido por otras razones. Clases que intervienen: Personalidad, Secuencia, Orden, Evento, Compilador, CompiladorPleo y Util.	Personalidad guarda en un fichero XML todas sus Secuencias con Eventos y Órdenes y pasa este fichero a Compilador. En este caso, CompiladorPleo realiza una traducción de dicho fichero a código de entrada para el compilador de Pleo y realiza la llamada a dicho compilador. El sistema detecta que la compilación no se ha podido llevar a cabo por algún motivo. Notifica al usuario que ha habido un error y pregunta si desea conservar el log.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

6.7.3 Pruebas de Usabilidad y Accesibilidad

PleoProg ha sido diseñado para que pueda ser utilizado por cualquier persona, con conocimientos o no, de informática. Las siguientes pruebas han sido diseñadas para comprobar que cualquier persona puede desenvolverse adecuadamente con la herramienta.

6.7.3.1 Pruebas de Usabilidad

Se ha diseñado un cuestionario que se completará con datos de las personas que realizan las pruebas sobre la herramienta y sobre cuestiones relacionadas con la herramienta.

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
¿Tiene conocimientos de programación?
1. Sí 2. No
¿Tiene conocimientos de robótica?
1. Sí 2. No
¿Qué valora más en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe donde está dentro de la herramienta?				
¿Existe ayuda para las funciones en caso de que tenga dudas?				
¿Le resulta sencillo el uso de la herramienta?				
¿Le resulta sencillo el sistema de interruptores para añadir los eventos y órdenes?				
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?				
¿El tiempo de respuesta de la herramienta es muy grande?				
¿Ha encontrado incoherencias entre lo deseado y lo representado en el diagrama?				
El archivo ejecutable ¿Es tal como usted ha creado en el diagrama?				
¿Encuentra incoherencias a la hora de cargar un archivo previamente guardado?				
¿Son las opciones lo suficientemente claras?				
¿Echa de menos alguna funcionalidad en la herramienta?				
¿Ha encontrado errores durante su utilización?				

Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es				
Los iconos e imágenes usados son				
Los colores empleados son				
El tamaño de los botones es				
El tamaño de los controles es				
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿Es adecuado el tamaño de las pantallas y cuadros mostrados?				
¿Cree que el programa está bien estructurado?				
Observaciones				
Cualquier comentario del usuario				

6.7.3.2 Cuestionario para el responsable de las pruebas

A continuación se muestra un cuestionario para el responsable de las pruebas en el que deberá valorar cada usuario que está probando la herramienta.

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	
El usuario consulta la ayuda	
Tiempo en realizar cada tarea	
El usuario se siente confundido	
El usuario cierra el programa por no saber continuar	
Tiempo en realizar una personalidad completa y probarla	
Errores leves cometidos	
Errores graves cometidos	

6.7.3.3 Actividades guiadas

A continuación se dan algunas actividades para que el usuario pueda probar en la aplicación:

- Realizar una personalidad completa
    - Añadir un evento
    - Añadir órdenes
    - Compilar
    - Probar
- Edición
    - Añadir eventos
    - Añadir órdenes
    - Eliminar eventos
    - Eliminar órdenes
    - Modificar eventos
    - Modificar órdenes
- Archivo
    - Nueva Personalidad
    - Guardar Personalidad
    - Cargar Personalidad

  - Cerrar Personalidad

- Modificación de preferencias
- Consulta de ayuda

6.7.3.4 Pruebas de Accesibilidad

Al tratarse de una aplicación de escritorio se han tenido en cuenta los siguientes aspectos a la hora de garantizar cierto nivel de accesibilidad:

- Atención en el diseño de los colores de la interfaz, estilo y tamaño de letra de todos los elementos mostrados.
- Atención en el diseño de los botones mediante iconos, que representen la función realizada.
- Uso de tooltips como fuentes de información alternativa.
- Botones de gran tamaño.
- Foco centrado en el primer elemento editable al abrir un cuadro.
- Botones por defecto de aceptación y cancelación en los cuadros.

6.7.4 Pruebas de Rendimiento

Para medir el rendimiento se medirá el consumo de diferentes recursos consumidos por la aplicación en diferentes momentos.

La memoria RAM se medirá en varios momentos para detectar posibles picos que no deberían darse. Los momentos en los que se medirá serán:

- Memoria RAM consumida en un momento inicial.
- Memoria RAM consumida con un diagrama extenso.
- Memoria RAM consumida en el compilado.

Para hacerse una idea del consumo de CPU se medirá el tiempo que lleva realizar varias tareas. Estas tareas son:

- Guardar un proyecto.
- Cargar un proyecto.
- Compilar.

En caso de detectar un consumo de memoria o tiempo de CPU excesivo se revisará la herramienta para minimizar el gasto de estos recursos.

Capítulo 7. Implementación del Sistema

7.1 Estándares y Normas Seguidos

Para el código desarrollado se han seguido las reglas generales de diseño SOLID para C# en su mayor parte, modificando algunas. Por ejemplo, SOLID dice que las propiedades deben escribirse utilizando la mayúscula para la primera letra, pero como se han utilizado propiedades automáticas se ha creído más conveniente aplicar la nomenclatura para los atributos de las clases (escribiendo la primera con minúscula) para que en el código resultante no quedara extraño.

Para el archivo XML generado se sigue el estándar XML 1.0.

7.2 Lenguajes de Programación

7.2.1 C#

C# es un lenguaje de programación orientado a objetos desarrollado por Microsoft para la plataforma .Net. Anders Hejelsberg, quien ya participara en la creación Turbo Pascal y J++, es el arquitecto principal de C#. El lenguaje fue estandarizado por primera vez en 2002 por la ECMA y en 2003 por ISO, en años posteriores sus revisiones fueron también estandarizadas.

Su sintaxis deriva de C/C++ y su modelo de objetos es parecido al de Java. C# es un lenguaje sencillo porque elimina elementos de otros lenguajes que no son necesarios en C#, el tamaño de los tipos básicos de datos es fijo e independiente del compilador, etc. Además soporta encapsulación, herencia y polimorfismo, añadiendo en algunos casos más opciones. En C# no es necesario liberar explícitamente memoria en el código, pues posee un recolector de basura automático. Una de sus principales características es que C# es un lenguaje seguro de tipos, pues incluye mecanismos de protección para evitar errores en los accesos de tipos de datos (conversión entre tipos compatibles, variables inicializadas, etc.).

El proyecto utiliza C# como lenguaje de programación debido a sus ventajas, algunas de las cuales han sido mencionadas anteriormente, la documentación disponible en internet y porque es uno de los lenguajes necesarios para el desarrollo en WPF.

Actualmente la última versión de C# es la 4.0, liberada en Abril de 2010 junto Microsoft Visual Studio 2010.

7.2.2 PAWN

PAWN es un lenguaje de programación creado específicamente para el desarrollo de aplicaciones para Pleo. Creado por Ugobe y liberado junto al PDK está dirigido a desarrolladores con experiencia que deseen programar para Pleo.

Es un lenguaje con una sintaxis muy parecida a C. PAWN se desarrolló para facilitar la rápida ejecución, estabilidad, simplicidad y reducido tamaño de las aplicaciones. Se encuentra en el propio firmware de Pleo para permitir la ejecución de código.

Una aplicación es una colección de diferentes recursos. Estos recursos pueden ser:

- Sonidos.
- Movimientos.
- Comandos. Tablas de movimientos asociados con información de estados, que describen cuando se debe ejecutar un movimiento.
- Scripts.
- Propiedades.

A través de PAWN se puede acceder a la API nativa de Pleo. Los módulos disponibles son los siguientes:

- ***Animation***. Funciones referentes a los comandos.
- ***Application***. Funciones referentes a la carga de aplicaciones.
- ***Attention***. Funciones referentes a la comunicación Pleo-Pleo.
- ***Camera*.** Funciones referentes a la cámara.
- ***common/comand_status***. Definiciones para el estado de comandos.
- ***common/message_type***. Definiciones para todos los tipos de menajes de log.
- ***default***. No define funciones pero sí valores para las versiones de la API de Pleo.
- ***Drive***. Funciones referentes al sistema de unidades.
- ***File***. Funciones para la entrada/salida de la memoria Data Flash o SD.
- ***Joint***. Funciones referentes a movimientos individuales.
- ***Log***. Funciones referentes al log, monitor y salida.
- ***Motion***. Funciones referentes a la reproducción de movimientos.
- ***pleo/active_syste*m**. Definiciones para los componentes de alto nivel.
- ***pleo/age***. Definiciones para posibles edades de Pleo.
- ***Pleo/joints***. Definiciones para todas las extremidades de Pleo.
- ***Pleo/limits***. Definiciones para los datos y el espacio de pila disponible.
- ***Pleo/mood***. Definiciones para los posibles valores de los estados de ánimo.
- ***Pleo/pose***. Definiciones para las posibles poses de Pleo.
- ***Pleo/propoerties***. Definiciones de las propiedades globales del sistema.
- ***Pleo/sensors***. Definiciones para todos los sensores disponibles en Pleo.
- ***Property***. Funciones referentes al sistema de Propiedades.
- ***Resource***. Funciones referentes a los archivos de recursos (URF).
- ***Script***. Funciones referentes a las máquinas virtuales de PAWN.
- ***Sensor***. Funciones referentes a los sensores.
- ***Sound***. Funciones referentes a la reproducción de sonidos.

- ***String***. Funciones referentes a las cadenas de PAWN.
- ***Time***. Funciones referentes a la hora.
- ***Util***. Funciones útiles variadas de utilidades.

En este proyecto se utilizan de las anteriores las siguientes:

- Log
- Script
- Sensor
- Sound
- Util
- Property
- Motion
- Joint

Para programar en PAWN es necesario saber que existen además 4 máquinas virtuales PAWN para permitir la multitarea.

- **SENSOR VM**. Cada vez que un sensor se lanza se ejecuta esta máquina que contiene las reacciones a posibles estímulos.

- **MAIN VM**. Se buscan la función *main* que se ejecuta indefinidamente.

- **BEHAVIOR VM**. Diseñada para ejecutar otro tipo de scripts arbitrarios. Busca la función *main* y la ejecuta.

- **USER VM**. Ejecuta scripts fuera del contexto de la aplicación actual.

En el proyecto actual, tratándose de crear una personalidad, las órdenes se ejecutan al recibir estímulos exteriores, por lo que se utilizará **SENSOR VM.**

A continuación se muestra un fragmento de código correspondiente al archivo sensors.p, que como se vio en el capítulo de aspectos teóricos será uno de los archivos que se dará como entrada al compilador.

Este fragmento tiene 4 funciones. La primera init() es la que se ejecuta sin necesidad de recibir ningún estímulo, la segunda on_sensor(…) se ejecuta cada vez que un sensor cambia su estado, la tercera es una simple función de espera y la cuarta se ejecuta al cerrar el script. Como se puede apreciar, y como se ha dicho antes, es muy parecido a C.

```
#pragma pack 1
#include <Log.inc>
#include <Script.inc>
#include <Sensor.inc>
#include <Sound.inc>
#include <Util.inc>
#include <Property.inc>
#include <Motion.inc>
#include <Joint.inc>

public init()
{
	print("sensors:init() enter\n");
	monitor_exec("joint neutral");
	monitor_exec("camera capture Imagen1.bmp bmp new");
	wait(7);
	print("sensors:init() exit\n");
}

public on_sensor(time, sensor_name: sensor, value)
{
	new name[32];
	sensor_get_name(sensor, name);
	printf("sensors:on_sensor(%d, %s, %d)\n", time, name, value);
	switch (sensor)
	{
		case SENSOR_HEAD:
			if (value==0)
			{
				monitor_exec("audio both Grabacion1.wav 10");
				monitor_exec("joint range 13 5 0 50 ");
			}
	}
	return true;
}

wait(value)
{
	new curTime=time();
	while(time()-curTime<(value*1000))
	{
	}
}

public close()
{
	print("sensors:close() enter\n");
	print("sensors:close() exit\n");
}
```

7.3 Herramientas y Programas Usados para el Desarrollo

7.3.1 Enterprise Architect

Para desarrollar todos los diagramas de la presente documentación se ha usado Enterprise Architect en su versión 7.8.850.

Se ha elegido este software ya que, por experiencia, se conocía su funcionamiento y porque habiendo probado otros software similares esta herramienta es la más completa.

7.3.2 Microsoft Visual Studio 2010

Microsoft Visual Studio es un entorno de desarrollo que permite la programación de forma nativa en los lenguajes Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET aunque existen extensiones para desarrollar en otros muchos lenguajes. La última versión incorpora .Net Framework 4.0 y una serie de mejoras en el aspecto visual del entorno.

Microsoft Visual Studio 2010 se utiliza en este proyecto para realizar la programación en WPF de la herramienta.

7.3.3 StyleCop

StyleCop es un plugin para Microsoft Visual Studio 2010 que analiza el código y muestra advertencias en aquellas líneas en las que no se estén siguiendo las líneas generales de diseño SOLID para C#.

Es una herramienta que ayuda a mantener un código ordenado, limpio y claro. La versión utilizada es la 4.3.3.0

7.3.4 Notepad ++

Para la vista y edición de muchos archivos es necesaria una herramienta tan rápida como el bloc de notas normal pero tan potente como un entorno de programación que reconozca las palabras reservadas. Para esta labor se ha usado Notepad ++ en su versión 5.5

7.3.5 HTML Help Workshop

Para la creación de un archivo de ayuda que se pudiera integrar adecuadamente con el resto de la aplicación, y en formato común se ha usado HTML Help Workshop de Microsoft en su versión 4.74.8702.

Se ha elegido dicho software por ser gratuito y de sencillo manejo.

7.3.6 GIMP

GIMP o GNU Image Manipulation Program (programa de manipulación de imágenes GNU) es una aplicación desarrollada para la edición y retoque de imágenes digitales. Inicialmente pensado para sistemas Unix, hoy en día tiene versiones tanto para Windows como para las principales distribuciones Linux. GIMP es actualmente una gran alternativa a Photoshop.

Para el retoque de imágenes que se usen durante todo el proyecto, tanto para las interfaces como para la documentación u otros recursos se ha usado GIMP por ser un editor de imágenes gratuito que satisface todas las necesidades requeridas para este proyecto.

La versión utilizada es la 2.6 para Windows.

7.3.7 Tortoise

A lo largo del desarrollo del proyecto se ha decidido usar un sistema SVN para mantener un control de cambios de todos los ficheros.

Se ha elegido Tortoise por la experiencia que se tenía con este programa y por tratarse el mejor programa encontrado para ello.

Menciona que se ha utilizado el servicio web unfuddle para esta labor.

7.3.8 WPF About (CS)

Para crear una ventana que muestre la información relativa al programa se ha descargado esta extensión para Visual Studio 2010 que genera una clase y una ventana que pueden ser llamadas para mostrar la información que se encuentra configurada en el proyecto.

Desarrollada por Pedro Silva (http://blogs.msdn.com/b/pedrosilva/)

La versión utilizada es la 0.5 y se ha modificado para ajustarla al proyecto.

Esta clase descargada se mostrará en todos los ámbitos de este proyecto, se recordará su procedencia y no se mostrará el código fuente en el anexo final para no confundir.

7.4 Creación del Sistema

7.4.1 Problemas Encontrados

En primer lugar, enfrentarse a un lenguaje (C#) y una tecnología (WPF) que nunca había utilizado fue un problema en algunos aspectos, pero dada la similitud de C# con Java el aprendizaje fue rápido.

Algunos problemas derivados de esta falta de conocimientos de C# fueron los siguientes:

- Interfaces, redimensionamiento.
- Utilización de carpetas y archivos temporales.
- Dibujado de los datos en pantalla y la interacción del usuario con estos (arrastrar)

Estos problemas mencionados fueron resueltos con perseverancia y gracias a dudas resueltas por otros usuarios en foros.

También, por falta de conocimientos, el diseño de interfaces mediante archivos XAML fue algo complicado al principio, pero con el tiempo se ha obtenido un nuevo punto de vista del desarrollo de aplicaciones donde más de un profesional tiene cabida, en este caso, un diseñador gráfico puede estar creando la interfaz mediante unas herramientas distintas de las que luego el programador usará para implementar la lógica.

El campo de la robótica nunca había sido tocado por mí, aunque sí que me atraía. Programar para un robot tiene ciertas limitaciones y es necesario conocer totalmente el hardware del que se dispone, investigar para Pleo llevó bastante tiempo.

7.4.2 Descripción Detallada de las Clases

A continuación se proporciona una descripción detallada de las clases. Los métodos y variables están ordenados alfabéticamente y las clases de un mismo paquete se presentan seguidas.

Destacar que, dado que se han utilizado propiedades automáticas en C#, la mayoría de los atributos no se han declarado manualmente, pero se tomaran como tal a todos los efectos.

Algunas de las clases, que tienen funcionalidad gráfica (presentar cuadros o ventanas) se complementan con los archivos de diseño de formato XAML, donde se declaran los controles.

7.4.2.1 MainWindow

Nombre	**Tipo**	**Descripción**	**Hereda de…**
MainWindow		Pantalla Principal de la aplicación. Siempre ejecutándose mientras la herramienta esté activa.	Window
Responsabilidades			
Número	**Descripción**		
1	Mostrar pantalla principal		
2	Controlar el estado de todos los elementos gráficos		
3	Mostrar pantalla inserción Evento		
4	Mostrar pantalla inserción órdenes		
5	Mostrar pantalla editar Eventos		
6	Mostrar pantalla editar Órdenes		
7	Eliminar Eventos		
8	Eliminar Órdenes		
9	Aumentar turno Órdenes		
10	Disminuir turno Órdenes		
11	Mostrar pantalla compilado		
12	Mostrar pantalla preferencias		
13	Mostrar ayuda		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private	int	botonOrden	void
private	void	buttonCam_Click	object sender, RoutedEventArgs e
private	void	buttonCompile_Click	object sender, RoutedEventArgs e
prívate	void	buttonDecrementarTurno_Click	object sender, RoutedEventArgs e
private	void	buttonEliminar_Click	object sender, RoutedEventArgs e
private	void	buttonEsp_Click	object sender, RoutedEventArgs e
prívate	void	buttonEvento_Click	object sender, RoutedEventArgs e
private	void	buttonMotion_Click	object sender,

			RoutedEventArgs e
private	void	buttonMov_Click	object sender, RoutedEventArgs e
prívate	void	buttonPosNormal_Click	object sender, RoutedEventArgs e
private	void	buttonRec_Click	object sender, RoutedEventArgs e
private	void	buttonRep_Click	object sender, RoutedEventArgs e
prívate	void	buttonSubirTurno_Click	object sender, RoutedEventArgs e
private	void	canvasPrncpl_MouseLeftButtonUp	object sender, mouseButtonEventArgs e
private	void	cargar_Click	object sender, RoutedEventArgs e
private	bool	cargarProyecto	string sRuta
private	void	cerrar_Click	object sender, RoutedEventArgs e
private	void	comboBoxEventoActual_SelectionChanged	object sender, SelectionChangedEventArgs e
private	void	gridPrincpl_KeyDown	object sender, KeyEventArgs e
private	void	guardar_Click	object sender, RoutedEventArgs e
private	bool	guardarProyecto	string sRuta
private	void	inicio	void
public	void	MainWindow	void
private	void	menuAcerca_Click	object sender, RoutedEventArgs e
private	void	menuAyuda_Click	object sender, RoutedEventArgs e
private	void	menuPagina_Click	object sender, RoutedEventArgs e
private	void	menuPreferencias_Click	object sender, RoutedEventArgs e
private	void	nuevo_Click	object sender, RoutedEventArgs e
private	void	reiniciarCanvas	void
private	void	resetBotones	ToggleButton t
private	void	salir_Click	object sender, RoutedEventArgs e
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		GestorDSL	gestor
private		string	sNombreProyecto
private		string	SrutaProyecto
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe MainWindow.xaml.cs			

7.4.2.2 Compilador

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Compilador	Abstracta	Sirve como clase general de la que heredaran todos los compiladores que se implementen, estos deben implementar sus métodos.	
Responsabilidades			
Número	**Descripción**		
1	Servir como clase general para que las especificas implementen sus métodos		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	void	Compilador	int iFormaVida, int iVFormaVida, string sRutaXML, string sRutaDestino, string sRutaCompilador, string sNombreProyecto, string sRutaTemp, StreamWriter swLog
public abstract	bool	compilar	void
public abstract	bool	rutaCompiladorValida	void
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
protected		int	iFormaVida
protected		int	iVFormaVida
protected		string	sRutaXML
protected		string	sRutaDestino
protected		string	sRutaCompilador
protected		string	sNombreProyecto
protected		string	sRutaTemp
protected		StreamWriter	swLog
Observaciones			
De esta clase heredan el resto de clases específicas de compiladores.			

7.4.2.3 CompiladorPleo

Nombre	Tipo	Descripción	Hereda de…
CompiladorPleo		Implementa los métodos específicos para compilar los programas para la forma de vida artificial Pleo	Compilador
Responsabilidades			
Número	**Descripción**		
1	Dar soporte para la compilación de un programa para la forma de vida Pleo		
2	Cargar la personalidad creada		
3	Generar los archivos necesarios para el compilador		
4	Llamar al compilador y pasar los archivos generados		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	void	CompiladorPleo	int iVFormaVida, string sRutaXML, string sRutaDestino, string sRutaCompilador, string sNombreProyecto, string sRutaTemp, StreamWriter swLog
public override	bool	compilar	void
private	void	crearArchivoP	void
private	void	crearArchivoUPF	void
private	void	crearCabcereaP	StreamWriter sW
private	void	crearCuerpoP	StreamWriter sW
private	bool	crearEsqueleto	void
private	void	insertarCaseAlzadoPosado	StreamWriter sW, XmlNodeList secuencia
private	void	insertarMotions	StreamWriter sW
private	void	insertarOrden	StreamWriter sW, XmlElement orden
private	void	insertarOrdenesCuerposP	StreamWriter sW
private	void	insertarOrdenesInicioP	StreamWriter sW
private	void	insertarSonidosUPF	StreamWriter sW
private	bool	llamarCompilador	void
public override	bool	rutaCompiladorValida	void
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		bool	bAlzadoPosado
private		bool	bEspera
private		bool	bMotions
private		bool	bSonidos

7.4.2.4 WindowCompilador

Nombre	**Tipo**	**Descripción**	**Hereda de…**
WindowCompilador		Pantalla de compilado	Window
Responsabilidades			
Número	**Descripción**		
1	Mostrar las preferencias de compilación		
2	Permitir editar las preferencias de compilación		
2	Iniciar el compilado		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private	void	agregarVersionesPleo	void
private	void	buttonCompiladorCancelar_Click	object sender, RoutedEventArgs e
private	void	buttonCompiladorCompilar_Click	object sender, RoutedEventArgs e
private	void	buttonCompiladorFilePDK_Click	object sender, RoutedEventArgs e
private	void	buttonCompiladorFileRuta_Click	object sender, RoutedEventArgs e
private	void	comboCompiladorFormaVida_SelectionChanged	object sender, SelectionChangedEventArgs e
private	void	Window_Closing	object sender, RoutedEventArgs e
public	void	WindowCompilador	string sRutaXML, string sNombreProyecto, string sRutaTemp
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		string	sNombreProyecto
private		string	sRutaCompilador
private		string	sRutaTemp
private		string	sRutaXML
private		StreamWriter	sWLog
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe MainWindow.xaml.cs			

7.4.2.5 Evento

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Evento		Almacena los datos de un evento	
Responsabilidades			
Número	**Descripción**		
1	Proporciona una estructura para almacenar datos de un evento		
2	Implementar método para comparar dos eventos		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public override	bool	Equals	Object evento
public	void	Evento	int iId
public	void	Evento	void
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		int	iId

7.4.2.6 Orden

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Orden	Abstracta	Clase general abstracta que sirva para las especificas como almacenamiento de Órdenes	
Responsabilidades			
Número	**Descripción**		
1	Proporciona una estructura de almacenamiento para las órdenes		
2	Servir como clase general para que las especificas implementen sus métodos		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public override	bool	Equals	Object o
public	void	Orden	int iID, int iTurno
public abstract	void	toXML	StreamWriter sW
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		int	iId
public		int	iTurno
Observaciones			
Clase abstracta, implementa Equals pero no toXML que debe implementar cada orden.			

7.4.2.7 Movimiento

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Movimiento		Sirve como estructura para almacenar una orden de movimiento	Orden
Responsabilidades			
Número	**Descripción**		
1	Proporciona una estructura de almacenamiento para la orden Movimiento		
2	Exportar la orden de la personalidad a un XML		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	void	Movimiento	int uId, int uTurno, int iGradosInicio, int iGradosFin, int iRepeticiones, int iMiembro, bool bRepetitivo
public override	void	toXML	StreamWriter sW
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		bool	bRepetitivo
public		int	iGradosFin
public		int	iGradosInicio
public		int	iMiembro
public		int	iRepeticiones

7.4.2.8 Neutro

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Neutro		Sirve como estructura para almacenar una orden de posición neutra	Orden
Responsabilidades			
Número	**Descripción**		
1	Proporciona una estructura de almacenamiento para la orden de posición neutra		
2	Exportar la orden de la personalidad a un XML		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	void	Neutro	int iId, int iTurno
public override	void	toXML	StreamWriter sW
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Observaciones			
Esta clase no tiene atributos, con los que hereda es suficiente.			

7.4.2.9 CapturaImagen

Nombre	**Tipo**	**Descripción**	**Hereda de…**
CapturaImagen		Sirve como estructura para almacenar la orden CapturaImagen	Orden
Responsabilidades			
Número	**Descripción**		
1	Proporciona una estructura de almacenamiento para la orden		
2	Exportar la orden de la personalidad a un XML		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	void	CapturaImagen	int Id, int iTurno, string sNombreArchivo
public override	void	toXML	StreamWriter sW
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		string	sNombreArchivo

7.4.2.10 CapturaSonido

Nombre	**Tipo**	**Descripción**	**Hereda de…**
CapturaSonido		Sirve como estructura para almacenar la orden CapturaSonido	Orden
Responsabilidades			
Número	**Descripción**		
1	Proporciona una estructura de almacenamiento para la orden		
2	Exportar la orden de la personalidad a un XML		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	void	CapturaSonido	int Id, int iTurno, string sNombreArchivo, int iSegundos
public override	void	toXML	StreamWriter sW
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		int	iSegundos
public		string	sNombreArchivo

7.4.2.11 ReproducirSonido

Nombre	**Tipo**	**Descripción**	**Hereda de…**
ReproducirSonido		Sirve como estructura para el almacenaje de la orden ReproducirSonido	Orden
Responsabilidades			
Número	**Descripción**		
1	Proporciona una estructura de almacenamiento para la orden		
2	Exportar la orden de la personalidad a un XML		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	void	ReproducirSonido	int iId, int iTurno, string sRutaArchivo
public override	void	toXML	StreamWriter sW
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		string	sRutaArchivo

7.4.2.12 Espera

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Espera		Sirve para como estructura para el almacenaje de la orden Espera	Orden
Responsabilidades			
Número	**Descripción**		
1	Proporciona una estructura de almacenamiento para la orden		
2	Exportar la orden de la personalidad a un XML		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	void	Espera	int iId, int iTurno, int iEspera
public override	void	toXML	StreamWriter sW
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		int	iEspera

7.4.2.13 Motion

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Motion		Sirve como estructura para el almacenaje de la orden Motion	Orden
Responsabilidades			
Número	**Descripción**		
1	Proporciona una estructura de almacenamiento para la orden Motion		
2	Exportar la orden de la personalidad a un XML		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	void	Motion	int iId, int iTurno, string sRutaAnimacion
public override	void	toXML	StreamWriter sW
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		string	sRutaAnimación

7.4.2.14 Personalidad

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Personalidad		Sirve como estructura de almacenamiento de la personalidad y para realizar operaciones sobre ella	
Responsabilidades			
Número	**Descripción**		
1	Almacenar la personalidad actual		
2	Editar la personalidad actual		
3	Exportar la personalidad a un XML		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	Secuencia	buscarSecuencia	Evento ev
public	Secuencia	buscarSecuencia	Orden orden
public	bool	editarSecuencia	Evento oldEv, Evento newEv
public	bool	eliminarEvento	Evento ev
public	List<Secuencia>	getSecuencias	void
public	Secuencia	insertarEvento	Evento ev
public	int	iSize	void
public	void	Personalidad	void
public	void	toXML	StreamWriter sW
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		int	iAcumulador
private		List<Secuencia>	lsSecuencias

7.4.2.15 Secuencia

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Secuencia		Sirve como estructura para almacenar una secuencia de la personalidad y permitir operaciones sobre ella	
Responsabilidades			
Número	**Descripción**		
1	Almacenar secuencias		
2	Editar secuencias		
3	Exportar la secuencia a XML		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	Orden	buscarOrden	int iTurno
public	void	editarEvento	Evento ev
public	bool	eliminarOrden	Orden orden
public	bool	existe	Orden orden
public	LinkedList<Orden>	getOrdenes	void
public	bool	insertarOrden	Orden orden
public	void	Secuencia	Evento ev
public	void	toXML	StreamWriter sW
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		Evento	evento
private		LinkedList<Orden>	lnkLsOrdenes

7.4.2.16 ControlEvento

Nombre	**Tipo**	**Descripción**	**Hereda de…**
ControlEvento		Clase utilizada para el correcto funcionamiento del control personalizado ControlEvento	UserControl
Responsabilidades			
Número	**Descripción**		
1	Representar un evento		
2	Representar de forma coherente el evento		
3	Permitir interacción con el usuario para mover y editar		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	void	ControlEvento	Secuencia secuencia, GestorDSL gestor, Point posControl
public	void	deseleccionar	void
private	void	inicializarDatos	void
public	void	seleccionar	void
private	void	userControl_MouseDoubleClick	object sender, MouseButtonEventArgs e
private	void	userControl_MouseDown	object sender, MouseButtonEventArgs e
private	void	userControl_MouseMove	object sender, MouseEventArgs e
private	void	userControl_MouseUp	object sender, MouseButtonEventArgs e
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		bool	bSeleccionado
private		GestorDSL	gestor
public		Line	lineaSig
public		Point	posControl
public		Secuencia	secuencia
public		Evento	evento
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe ControlEvento.xaml.cs			

7.4.2.17 ControlOrden

Nombre	**Tipo**	**Descripción**	**Hereda de…**
ControlOrden		Clase utilizada para el correcto funcionamiento del control personalizado ControlOrden	UserControl
Responsabilidades			
Número	**Descripción**		
1	Representar una orden		
2	Representar de forma coherente la orden		
3	Permitir interacción con el usuario para mover y editar		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	void	ControlOrden	Secuencia secuencia, Orden orden, GestorDSL gestor, Point posControl
public	void	deseleccionar	void
private	void	inicializarDatos	void
private	void	mostrarMiembro	int iMiembro
private	void	mostrarObservaciones	string sDetalles
private	void	ocultarMiembro	void
private	void	ocultarObservaciones	void
public	void	seleccionar	void
private	void	userControl_MouseDoubleClick	object sender, MouseButtonEventArgs e
private	void	userControl_MouseDown	object sender, MouseButtonEventArgs e
private	void	userControl_MouseMove	object sender, MouseEventArgs e
private	void	userControl_MouseUp	object sender, MouseButtonEventArgs e
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		bool	bSeleccionado
public		Evento	evento
private		GestorDSL	gestor
public		Line	lineaSig
public		Point	posControl
public		Secuencia	secuencia
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe ControlOrden.xaml.cs			

7.4.2.18 ScrollableCanvas

Nombre	**Tipo**	**Descripción**	**Hereda de…**
ScrollableCanvas		Redefine canvas para que se ajuste a las necesidades del proyecto	Canvas
Responsabilidades			
Número	**Descripción**		
1	Redefinir método MeasureOverride		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
protected override	Size	MeasureOverride	Size constraint
public	void	ScrollableCanvas	void
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Observaciones			
Todos los atributos que necesita ya se encuentran en la clase Canvas			

7.4.2.19 GestorDSL

Nombre	Tipo	Descripción	Hereda de…
GestorDSL		Sirve para gestionar los elementos gráficos mostrados en MainWindow	

Responsabilidades

Número	Descripción
1	Mantener en memoria la Personalidad actual
2	Dibujar los elementos gráficos apropiados que representen los eventos y ordenes que el usuario ha añadido
3	Proporciona funcionalidad para guardar y cargar una personalidad
4	Subir el turno y decrementar el turno de las órdenes

Métodos

Acceso | Modo	Tipo de Retorno	Nombre	Parámetros y tipos
public	void	abrirPersonalidad	string sRuta
public	void	añadirElementoComboBox	int iEv
public	ControlEvento	buscarControlEvento	Evento ev
public	ControlOrden	buscarControlOrden	Orden orden
public	void	decrementarTurno	ControlOrden cO
public	void	dibujarControl	Point ptDib, Secuencia sec, Object accion
private	void	dibujarLinea	Point ptOrigen, bool bOrEv, Point ptDestino, bool bDesEv, Line linea
public	void	dibujarPersonalidad	void
public	void	eliminarControlInterfaz	object control
public	void	eliminarElementoComboBox	int iEv
private	bool	existeElementoComboBox	string sEvento
public	void	GestorDSL	Canvas canvas, ComboBox cmbEvActual
public	void	guardarPersonalidad	string sRuta
public	void	incrementarTurno	ControlOrden cO
public	Line	nuevaLinea	Point ptOrigen, bool bOrEv, Point ptDestino, bool bDesEv
public	void	redibujarLinea	ControlOrden cO, Point ptOrigen
public	void	redibujarLinea	ControlEvento cE, Point ptOrigen
public	void	seleccionarElementoComboBox	int iEv

Atributos

Acceso	Modo	Tipo o Clase	Nombre
public		Canvas	canvas
public		ComboBox	cmbEvActual

public		ControlEvento	eventoActual
public		ControlEvento	eventoSeleccionado
public		ControlOrden	ordenSeleccionada
public		Personalidad	personalidad
private		LinkedList<ControlOrden>	lnkLsOrdenes
private		LinkedList<ControlEvento>	lnkLsEventos

7.4.2.20 WindowEvento

Nombre	**Tipo**	**Descripción**	**Hereda de…**
WindowEvento		Pantalla que posibilita elegir un evento para crear o para modificar	Window
Responsabilidades			
Número	**Descripción**		
1	Mostrar todos los eventos disponibles		
2	Modificar un evento		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private	void	buttonAceptarEvento_Click	object sender, RoutedEventArgs e
private	void	buttonCancelarEvento_Click	object sender, RoutedEventArgs e
public	void	WindowEvento	Evento ev, GestorDSL gestor
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		Evento	ev
private		GestorDSL	gestor
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe WindowEvento.xaml.cs			

7.4.2.21 WindowMovimiento

Nombre	**Tipo**	**Descripción**	**Hereda de…**
WindowMovimiento		Pantalla que permite crear una orden de movimiento o modificarla	Window
Responsabilidades			
Número	**Descripción**		
1	Mostrar las opciones posibles para cada orden de movimiento		
2	Modificar las opciones de la orden de movimiento		
3	Verificar que las opciones introducidas son correctas		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private	bool	anguloValidoPleo	int iMiembro, int iGradosFinal, bool bRepetitivo, int iGradosInicioa
private	void	buttonAceptarMovimiento_Click	object sender, RoutedEventArgs e
private	void	buttonCancelarMovimiento_Click	object sender, RoutedEventArgs e
private	void	RBRepetitivo_Checked	object sender, RoutedEventArgs e
private	void	RBUnico_Checked	object sender, RoutedEventArgs e
private	void	textBoxPos_KeyDown	object sender, KeyEventArgs e
private	void	textBoxPosfinal_KeyDown	object sender, KeyEventArgs e
private	void	textBoxRep_KeyDown	object sender, KeyEventArgs e
public	void	WindowMovimiento	Movimiento o
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		Orden	orden
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe WindowMovimiento.xaml.cs			

7.4.2.22 WindowCapturaSonido

Nombre	**Tipo**	**Descripción**	**Hereda de…**
WindowCapturaSonido		Pantalla que permite crear una orden de CapturaSonido o modificarla	Window
Responsabilidades			
Número	**Descripción**		
1	Mostrar el nombre del archivo que se creará		
2	Modificar el nombre del archivo de la orden CapturaSonido		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private	void	buttonAceptar_Click	object sender, RoutedEventArgs e
private	void	buttonCancelar_Click	object sender, RoutedEventArgs e
private	void	textBoxNomArchivo_Click	object sender, KeyEventArgs e
private	void	textBoxTiempoSonido_Click	object sender, KeyEventArgs e
public	void	WindowCapturaSonido	Orden o
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		CapturaSonido	cS
private		int	iSegundos
private		string	sNombre
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe WindowCapturaSonido.xaml.cs			

7.4.2.23 WindowTiempoEspera

Nombre	**Tipo**	**Descripción**	**Hereda de…**
WindowTiempoEspera		Pantalla que permite crear una orden de TiempoEspera o modificarla	Window
Responsabilidades			
Número	**Descripción**		
1	Mostrar la duración del tiempo en espera de la orden		
2	Modificar el tiempo en espera de la orden		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private	void	buttonAceptar_Click	object sender, RoutedEventArgs e
private	void	buttonCancelar_Click	object sender, RoutedEventArgs e
private	void	textBoxTiempoEspera_KeyDown	object sender, KeyEventArgs e
public	void	WindowTiempoEspera	Espera eEsp
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		Espera	eEsp
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe WindowTiempoEspera.xaml.cs			

7.4.2.24 WindowNombreArchivo

Nombre	**Tipo**	**Descripción**	**Hereda de…**
WindowNombreArchivo		Pantalla para introducir el nombre de un archivo	Window
Responsabilidades			
Número	**Descripción**		
1	Mostrar la ruta de un archivo		
2	Modificar la ruta de un archivo		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private	void	buttonAceptar_Click	object sender, RoutedEventArgs e
private	void	buttonCancelar_Click	object sender, RoutedEventArgs e
private	void	textBoxNomArchivo_KeyDown	object sender, KeyEventArgs e
public	void	WindowNombreArchivo	string sNombre
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		string	sNombre
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe WindowNombreArchivo.xaml.cs			

7.4.2.25 WindowNombreArchivoOrden

Nombre	**Tipo**	**Descripción**	**Hereda de…**
WindowNombreArchivoOrden		Pantalla para introducir la ruta de un archivo que se adjuntara a una orden	Window
Responsabilidades			
Número	**Descripción**		
1	Mostrar la ruta de un archivo que se adjuntará a una orden		
2	Modificar la ruta de un archivo que se adjuntará a una orden		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private	void	buttonAceptar_Click	object sender, RoutedEventArgs e
private	void	buttonCancelar_Click	object sender, RoutedEventArgs e
private	void	textBoxNomArchivo_KeyDown	object sender, KeyEventArgs e
public	void	WindowNombreArchivoOrden	string sNombre
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		Orden	orden
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe WindowNombreArchivoOrden.xaml.cs			

7.4.2.26 WindowOpcionesArchivo

Nombre	**Tipo**	**Descripción**	**Hereda de…**
WindowOpcionesArchivo		Pantalla para que el usuario seleccione si desea crear una nueva personalidad o cargarla	Window
Responsabilidades			
Número	**Descripción**		
1	Permitir al usuario elegir entre crear un nuevo proyecto o cargarlo		
2	Seleccionar la ruta del archivo a cargar		
3	Insertar nombre del nuevo proyecto		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private	void	buttonAceptar_Click	object sender, RoutedEventArgs e
private	void	buttonExplorar_Click	object sender, RoutedEventArgs e
private	void	buttonSalir_Click	object sender, RoutedEventArgs e
private	void	RBCargar_Checked	object sender, RoutedEventArgs e
private	void	RBNuevo_Checked	object sender, RoutedEventArgs e
public	void	WindowOpcionesArchivo	void
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		string	sOpcion
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe WindowOpcionesArchivo.xaml.cs			

7.4.2.27 WindowAcerca

Nombre	**Tipo**	**Descripción**	**Hereda de…**
WindowAcerca		Pantalla que muestra información acerca de la herramienta	Window
Responsabilidades			
Número	**Descripción**		
1	Mostrar información acerca de la herramienta y el autor		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private	string	CalculatePropertyValue	string propertyName, string xpathQuery
protected virtual	string	GetLogicalResourcesString	string xpathQuery
public	void	WindowAcerca	void
public	void	WindowAcerca	Window parent
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
public		string	Company
public		string	Copyright
public		string	Description
public		string	LinkText
public		string	LinkUri
public		string	Product
public		string	ProductTitle
protected	virtual	XmlDocument	ResourceXmlDocument
public		string	Version
Observaciones			
Esta clase ha sido creada por el plugin WPF About Box (CS) por Pedro Silva, pero ha sido modificada para adaptarse al proyecto			

7.4.2.28 WindowPreferencias

Nombre	**Tipo**	**Descripción**	**Hereda de…**
WindowPreferencias		Pantalla que sirve para mostrar las preferencias del programa	Window
Responsabilidades			
Número	**Descripción**		
1	Mostrar preferencias del programa		
2	Editar preferencias del programa		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private	void	buttonAceptar_Click	object sender, RoutedEventArgs e
private	void	buttonCancelar_Click	object sender, RoutedEventArgs e
private	void	buttonExplorar_Click	object sender, RoutedEventArgs e
private	void	buttonExplorarCompilador_Click	object sender, RoutedEventArgs e
private	void	buttonExplorarDestino_Click	object sender, RoutedEventArgs e
private	void	CBGuardarLog_Checked	object sender, RoutedEventArgs e
public	void	WindowPreferencias	void
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Observaciones			
Clase que implementa funcionalidad gráfica, aquí sólo se describe WindowPreferencias.xaml.cs			

7.4.2.29 Util

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Util	Estática	Almacena variables que sirven para identificar partes, movimientos, órdenes y eventos	

Responsabilidades	
Número	**Descripción**
1	Proveer acceso a las variables que identifican partes, movimientos, órdenes y eventos

Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public static	int	getCodigoEvento	string sEvento
public static	string	getNombreEvento	int iIdEvento
public static	string	getNombreEventoCompiladorPleo	int iIdEvento
public static	string	getNombreMiembro	int iMiembro

Atributos

Acceso	**Modo**	**Tipo o Clase**	Nombre
public	const	int	altoControlEvento
public	const	int	altoControlOrden
public	const	int	anchoControlEvento
public	const	int	anchoControlOrden
public	const	int	capturaFoto
public	const	int	capturaSonido
public	const	int	eventoAbuso
public	const	int	eventoAgitado
public	const	int	eventoAlzado
public	const	int	eventoFilo
public	const	int	eventoInterruptorExtremidadDelanteraDerecha
public	const	int	eventoInterruptorExtremidadDelanteraIzquierda
public	const	int	eventoInterruptorExtremidadTraseraDerecha
public	const	int	eventoInterruptorExtremidadTraseraIzquierda
public	const	int	eventoObjetoBoca
public	const	int	eventoObstaculo
public	const	int	eventoOtroPleo
public	const	int	eventoPosado
public	const	int	eventoSonido
public	const	int	eventoStart
public	const	int	eventoTactilCabeza
public	const	int	eventoTactilCuello
public	const	int	eventoTactilEspalda
public	const	int	eventoTactilExtremidadDelanteraDerecha
public	const	int	eventoTactilExtremidadDelanteraIzquierda
public	const	int	eventoTactilExtremidadTraseraDerecha
public	const	int	eventoTactilExtremidadTraseraIzquierda
public	const	int	eventoTactilMenton
public	const	int	eventoVariacionLuz

public	const	int	eventoWakeUp
public	const	int	movCaderaDerecha
public	const	int	movCaderaIzquierda
public	const	int	movCodoDerecho
public	const	int	movCodoIzquierdo
public	const	int	movColaHorizontal
public	const	int	movColaVertical
public	const	int	movCuelloHorizontal
public	const	int	movCuelloVertical
public	const	int	movHombroDerecho
public	const	int	movHombroIzquierdo
public	const	int	movOjos
public	const	int	movRodillaDerecha
public	const	int	movRodillaIzquierda
public	const	int	movTorso
public	const	int	posNeutra
public	const	int	repSonido
public	const	int	tiempoEspera

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias

A continuación se detallan las pruebas unitarias realizadas y el resultado de dichas pruebas.

Caso de Uso 1: Insertar Evento		
Prueba	**Resultado Esperado**	**Resultado**
Guardar un evento cualquiera. Clases que intervienen: Personalidad, Secuencia y Evento y Util.	Se guarda correctamente un evento nuevo. La clase Personalidad posee una secuencia más.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	No se guarda ningún evento.	Satisfactorio

Caso de Uso 2:Eliminar Evento		
Prueba	**Resultado Esperado**	**Resultado**
Eliminar un evento existente. Clases que intervienen: Personalidad, Secuencia y Evento y Util.	Se elimina correctamente un evento. La clase Personalidad posee una secuencia menos.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	No se elimina ningún evento.	Satisfactorio

Caso de Uso 3: Insertar Orden		
Prueba	**Resultado Esperado**	**Resultado**
Guardar una orden cualquiera. Clases que intervienen: Secuencia, Orden y Util.	Se guarda correctamente una orden nueva. La clase Secuencia tiene una orden más.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	No se guarda ninguna orden.	Satisfactorio

Caso de Uso 4: Modificar Orden		
Prueba	**Resultado Esperado**	**Resultado**

Se modifica una orden existente. Clases que intervienen: Secuencia, Orden y Util.	Se modifican los datos de una orden. La clase Secuencia posee la misma orden pero modificada.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	El sistema permanece sin cambios.	Satisfactorio

Caso de Uso 5: Eliminar Orden		
Prueba	**Resultado Esperado**	**Resultado**
Eliminar una orden existente. Clases que intervienen: Secuencia, Orden y Util.	Se elimina una orden. La clase Secuencia posee una orden menos.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	El sistema permanece sin cambios.	Satisfactorio

Caso de Uso 6: Aumentar Turno		
Prueba	**Resultado Esperado**	**Resultado**
Aumentar turno de una orden. Clases que intervienen: Secuencia, Orden y Util.	La orden tiene un turno mayor. La clase Secuencia modifica dos órdenes, la orden que modifica su turno y la orden siguiente en la secuencia que también modifica su turno.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	El sistema permanece sin cambios.	Satisfactorio

Caso de Uso 7: Disminuir Turno		
Prueba	**Resultado Esperado**	**Resultado**
Disminuir turno de una orden Clases que intervienen: Secuencia, Orden y Util.	La orden tiene un turno menor. La clase Secuencia modifica dos órdenes, la orden que modifica su turno y la orden anterior en la secuencia que también modifica su turno.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	El sistema permanece sin cambios.	Satisfactorio

8.2 Pruebas de Integración y del Sistema

Caso de Uso 1: Insertar Evento		
Prueba	**Resultado Esperado**	**Resultado**
Insertar un evento no existente. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Evento y Util.	La clase Personalidad posee una nueva instancia de la clase Secuencia más. La clase Secuencia almacena correctamente los datos del evento introducido. La clase GestorDSL crea y dibuja el control apropiado en el área de trabajo.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Insertar un evento repetido. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Evento y Util.	El sistema detecta que dicho evento ya existe dentro de Personalidad y notifica dicha incidencia al usuario para que elija otro Evento.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	El sistema permanece sin cambios.	Satisfactorio

Caso de Uso 2:Eliminar Evento		
Prueba	**Resultado Esperado**	**Resultado**
Eliminar un evento sin órdenes. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Evento y Util.	La clase Personalidad posee una Secuencia menos. GestorDSL busca y elimina de la interfaz el control que representa dicho evento.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Eliminar un evento con órdenes. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Evento y Util.	El sistema detecta que se intenta borrar un evento que posee órdenes. Notifica al usuario que si continua borrará también dichas órdenes. En caso afirmativo, Personalidad posee una Secuencia menos y GestorDSL busca y elimina de la interfaz el control que representa dicho evento y las órdenes que posee.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	El sistema permanece sin cambios.	Satisfactorio

Caso de Uso 3: Insertar Orden		
Prueba	**Resultado Esperado**	**Resultado**
Insertar una orden existiendo un evento. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	La clase Personalidad asigna a Secuencia dicha Orden y Secuencia posee una Orden más. GestorDSL crea y dibuja el control apropiado en el área de trabajo.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Insertar una orden sin que exista ningún evento o que no se encuentre ninguno activo. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Evento, Orden y Util.	El sistema detecta que no existe ningún evento o que ninguno se encuentra activo. Notifica al usuario que añada o seleccione algún evento de los disponibles y lo vuelva a intentar. No se añade ninguna Orden a ninguna Secuencia.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Insertar una orden con opciones incorrectas. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	El sistema detecta que las opciones introducidas no son correctas. Notifica al usuario que revise las opciones introducidas. No se añade ninguna Orden a ninguna Secuencia.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Insertar una orden que incluya un archivo de recurso (sonido o animación) superior a 1 MB. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	El sistema detecta que se intenta adjuntar un archivo con un tamaño superior a 1MB. Notifica al usuario que el archivo es demasiado grande. No se añade ninguna Orden a ninguna Secuencia.	Satisfactorio
Prueba	**Resultado Esperado**	
Cancelar la Operación	El sistema permanece sin cambios.	

Caso de Uso 4: Modificar Orden		
Prueba	**Resultado Esperado**	**Resultado**
Modificar una orden con opciones	Personalidad realiza los cambios en la Orden de la Secuencia y GestorDSL realiza los cambios	Satisfactorio

correctas. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	gráficamente del control que representa dicha Orden en el área de trabajo.	
Prueba	**Resultado Esperado**	**Resultado**
Modificar una orden con opciones incorrectas Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	El sistema detecta que las opciones de la Orden no son correctas. Notifica al usuario para que lo vuelva a intentar o cancele dichos cambios. La Orden de la Secuencia no se modifica.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	El sistema permanece sin cambios.	Satisfactorio

Caso de Uso 5: Eliminar Orden		
Prueba	**Resultado Esperado**	**Resultado**
Eliminar una orden existente. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	La Personalidad solicita a Secuencia que se elimine una Orden. Secuencia posee una Orden menos. GestorDSL elimina el control pertinente del área de trabajo.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	El sistema permanece sin cambios.	Satisfactorio

Caso de Uso 6: Aumentar Turno		
Prueba	**Resultado Esperado**	**Resultado**
Se aumenta el turno de una orden que no esté en último lugar. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	La clase Personalidad solicita a Secuencia cambiar el turno de la Orden seleccionada y de la Orden siguiente intercambiando los turnos. GestorDSL intercambia el lugar y el turno de la Orden seleccionada y de la Orden siguiente.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Se aumenta el turno de una orden que se encuentra en último lugar.	El sistema detecta que la Orden seleccionada se encuentra en último lugar dentro de la Secuencia y no posee órdenes posteriores por lo que no se realiza ningún tipo de cambio.	Satisfactorio

Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.		
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	El sistema permanece sin cambios.	Satisfactorio

Caso de Uso 7: Disminuir Turno		
Prueba	**Resultado Esperado**	**Resultado**
Se disminuye el turno de una orden que no esté en primer lugar. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	La clase Personalidad solicita a Secuencia cambiar el turno de la Orden seleccionada y de la Orden anterior intercambiando los turnos. GestorDSL intercambia el lugar y el turno de la Orden seleccionada y de la Orden anterior.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Se disminuye el turno de una orden que se encuentra en primer lugar. Clases que intervienen: GestorDSL, Personalidad, Secuencia, Orden y Util.	El sistema detecta que la Orden seleccionada se encuentra en primer lugar dentro de la Secuencia y no posee órdenes anteriores por lo que no se realiza ningún tipo de cambio.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	El sistema permanece sin cambios.	Satisfactorio

Caso de Uso 8: Compilar		
Prueba	**Resultado Esperado**	**Resultado**
Compilar un programa válido. Clases que intervienen: Personalidad, Secuencia, Orden, Evento, Compilador, CompiladorPleo y Util.	Personalidad guarda en un fichero XML todas sus Secuencias con Eventos y Órdenes y pasa este fichero a Compilador. En este caso, CompiladorPleo realiza una traducción de dicho fichero a código de entrada para el compilador de Pleo y realiza la llamada a dicho compilador. En la ruta destino se crea un archivo ejecutable por Pleo.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Compilar un programa sin eventos.	El sistema detecta que la Personalidad no tiene ninguna Secuencia. Notifica al usuario que no se	Satisfactorio

Clases que intervienen: Personalidad, Secuencia, Orden, Evento, Compilador, CompiladorPleo y Util.	puede compilar un programa vacio. No se realiza la compilación.	
Prueba	**Resultado Esperado**	**Resultado**
Compilar un programa cuando la ruta del compilador es errónea. Clases que intervienen: Personalidad, Secuencia, Orden, Evento, Compilador, CompiladorPleo y Util.	El sistema detecta que la carpeta del compilador especificada no es correcta. Notifica al usuario y le sugiere cambiar dicha ruta. No se realiza ninguna compilación.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Compilar un programa cuando la ruta de destino es errónea. Clases que intervienen: Personalidad, Secuencia, Orden, Evento, Compilador, CompiladorPleo y Util.	El sistema detecta que la ruta de destino es errónea. Notifica al usuario para que cambie dicha ruta. No se realiza ninguna compilación.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Compilar un programa no válido por otras razones. Clases que intervienen: Personalidad, Secuencia, Orden, Evento, Compilador, CompiladorPleo y Util.	Personalidad guarda en un fichero XML todas sus Secuencias con Eventos y Órdenes y pasa este fichero a Compilador. En este caso, CompiladorPleo realiza una traducción de dicho fichero a código de entrada para el compilador de Pleo y realiza la llamada a dicho compilador. El sistema detecta que la compilación no se ha podido llevar a cabo por algún motivo. Notifica al usuario que ha habido un error y pregunta si desea conservar el log.	Satisfactorio
Prueba	**Resultado Esperado**	**Resultado**
Cancelar la Operación	El sistema permanece sin cambios.	Satisfactorio

8.3 Pruebas de Usabilidad y Accesibilidad

8.3.1 Pruebas de Usabilidad

8.3.1.1 Cuestionarios

A continuación se mostrarán los resultados de realizar los cuestionarios desarrollados en diseño. Se mostrarán las preguntas formuladas y la cantidad de veces respondida.

Para realizar los cuestionarios se ha entrevistado a 3 personas de diferentes rangos de edad y conocimientos informáticos.

Se ha permitido la elección múltiple en las preguntas cuyas respuestas no fueran excluyentes.

¿Usa un ordenador frecuentemente?	**Resultados**
1. Todos los días	1/3
2. Varias veces a la semana	1/3
3. Ocasionalmente	1/3
4. Nunca o casi nunca	0/3
¿Qué tipo de actividades realiza con el ordenador?	**Resultados**
1. Es parte de mi trabajo o profesión	0/3
2. Lo uso básicamente para ocio	2/3
3. Solo empleo aplicaciones estilo Office	1/3
4. Únicamente leo el correo y navego ocasionalmente	1/3
¿Ha usado alguna vez software como el de esta prueba?	**Resultados**
1. Sí, he empleado software similar	0/3
2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares	0/3
3. No, nunca	3/3
¿Tiene conocimientos de programación?	**Resultados**
1. Sí	1/3
2. No	2/3
¿Tiene conocimientos de robótica?	**Resultados**
1. Sí	0/3
2. No	3/3

¿Qué valora más en un programa?	**Resultados**
1. Que sea fácil de usar	1/3
2. Que sea intuitivo	1/3
3. Que sea rápido	3/3
4. Que tenga todas las funciones necesarias	3/3

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe donde está dentro de la herramienta?	2/3	1/3	0/3	0/3
¿Existe ayuda para las funciones en caso de que tenga dudas?	1/3	1/3	1/3	0/3
¿Le resulta sencillo el uso de la herramienta?	3/3	0/3	0/3	0/3
¿Le resulta sencillo el sistema de interruptores para añadir los eventos y órdenes?	0/3	2/3	1/3	0/3
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?	2/3	1/3	0/3	0/3
¿El tiempo de respuesta de la herramienta es muy grande?	0/3	0/3	1/3	2/3
¿Ha encontrado incoherencias entre lo deseado y lo representado en el diagrama?	0/3	0/3	0/3	3/3
El archivo ejecutable ¿Es tal como usted ha creado en el diagrama?	3/3	0/3	0/3	0/3
¿Encuentra incoherencias a la hora de cargar un archivo previamente guardado?	0/3	0/3	0/3	3/3
¿Son las opciones lo suficientemente claras?	2/3	1/3	0/3	0/3
¿Echa de menos alguna funcionalidad en la herramienta?	0/3	0/3	0/3	3/3
¿Ha encontrado errores durante su utilización?	0/3	0/3	0/3	3/3
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es	2/3	1/3	0/3	0/3
Los iconos e imágenes usados son	1/3	2/3	0/3	0/3
Los colores empleados son	2/3	1/3	0/3	0/3
El tamaño de los botones es	3/3	0/3	0/3	0/3
El tamaño de los controles es	3/3	0/3	0/3	0/3
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?		3/3	0/3	0/3
¿El diseño de las pantallas es claro y atractivo?		2/3	0/3	1/3
¿Es adecuado el tamaño de las pantallas y		2/3	0/3	1/3

cuadros mostrados?			
¿Cree que el programa está bien estructurado?	3/3	0/3	0/3

8.3.1.2 Cuestionario para el responsable de las pruebas

A continuación se muestra un resumen de las notas tomadas en los cuestionarios para las pruebas.

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	Tardan un momento en entender el funcionamiento de los botones
El usuario consulta la ayuda	Todos los usuarios han consultado la ayuda en uno u otro momento
Tiempo en realizar cada tarea	El tiempo utilizado en crear un diagrama la primera vez es alto, pero a partir de este primer diagrama el resto de veces lo realizan rápidamente
El usuario se siente confundido	Solo al comienzo
El usuario cierra el programa por no saber continuar	Nunca
Tiempo en realizar una personalidad completa y probarla	La primera vez, mucho, el resto rápidamente.
Errores leves cometidos	Intentar arrastrar los botones al área de trabajo.
Errores graves cometidos	Todos los usuarios intentan insertar una orden sin haber insertado un evento previamente, en cuanto la herramienta les advierte que primero se debe insertar el evento, no se vuelve a repetir el error.

8.3.1.3 Actividades guiadas

Las actividades guiadas a desarrollar fueron las que se recomendaron al usuario para que realizara.

8.3.2 Pruebas de Accesibilidad

A continuación se muestra un cuadro con las pruebas de accesibilidad desarrolladas en el diseño e indicando si se han implementando todas.

Aspecto	**Implementado**
Atención en el diseño de los colores de la interfaz, estilo y tamaño de letra de todos los elementos mostrados.	Sí.
Atención en el diseño de los botones mediante iconos, que representen la función realizada.	Sí.
Uso de tooltips como fuentes de información alternativa.	Sí.
Botones de gran tamaño.	Sí.
Foco centrado en el primer elemento editable al abrir un cuadro.	Sí.
Botones por defecto de aceptación y cancelación en los cuadros.	Sí.

8.4 Pruebas de Rendimiento

Como se detallo en el diseño, se han llevado a cabo las siguientes pruebas para que sirvan como referencia de rendimiento de la herramienta.

Momento para medir RAM	**Cantidad de Memoria**
Memoria RAM consumida en un momento inicial	40720 KB
Memoria RAM consumida con un diagrama extenso (más de 100 órdenes)	64672 KB
Memoria RAM consumida en el compilado	No cambia
Prueba	**Tiempo**
Guardar un proyecto	Menos de 1 segundo
Cargar un proyecto	Menos de 1 segundo
Compilar	Menos de 3 segundos

Por lo que se aprecia de las anteriores pruebas el rendimiento de la herramienta desarrollada es satisfactorio. El consumo de memoria se encuentre dentro de los límites normales y el tiempo en realizar las operaciones más complejas se mantiene muy bajo.

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

Instalar PleoProg es una tarea sencilla que no requiere grandes conocimientos. En esta guía se muestra como instalar teniendo en cuenta ciertos aspectos.

PleoProg se ejecuta sobre .Net Framework 4, es por ello que los requisitos mínimos son los que señala dicha tecnología, estos son:

- Procesador, 1GHz.
- RAM, 512 Mb,
- Espacio en Disco Duro, 850 Mb para sistemas de 32 bits y 2GB para sistemas de 64 bits.
- Windows instalado, la mayoría de versiones recientes tienen soporte.

Además para desarrollar para Pleo es necesario que descargue el PDK, de cara a que la herramienta pueda hacer uso del compilador. Para ello visite http://www.pleoworld.com/downloads/pdk.aspx

Para instalar PleoProg ejecute el archivo Setup.exe que verá dentro de la carpeta de instalación.

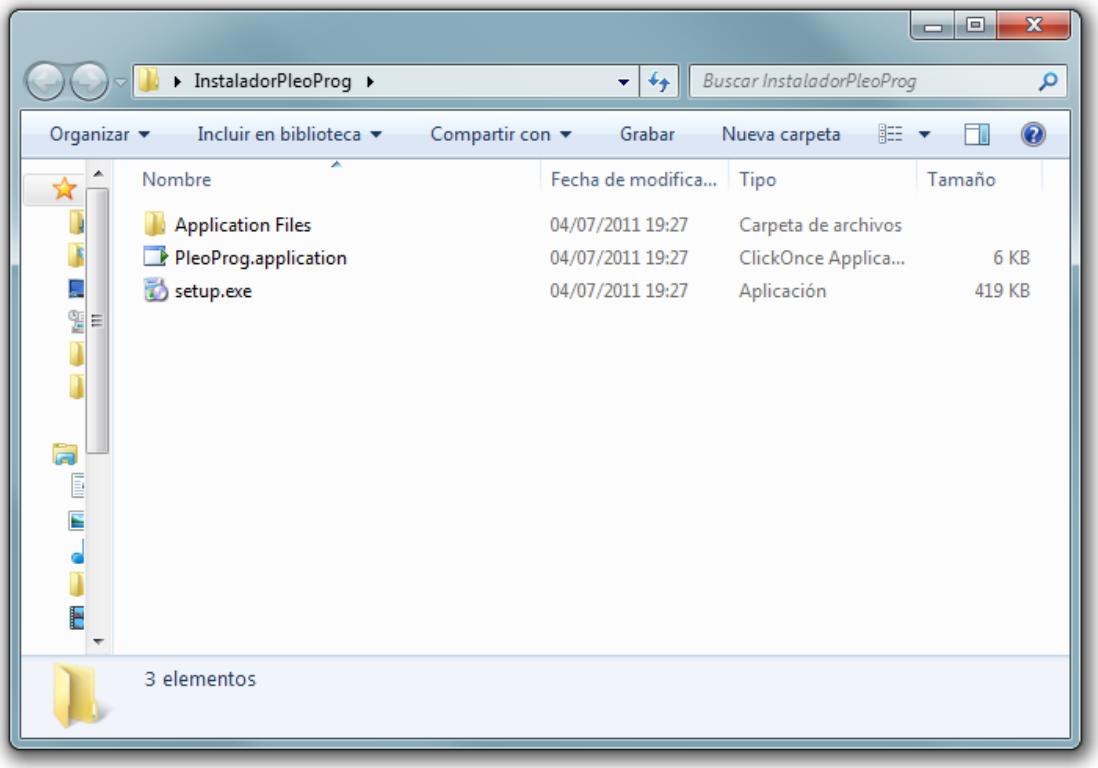


Figura 9.1 Carpeta raíz instalación

Al ejecutar verá una advertencia, haga clic en “Install”.

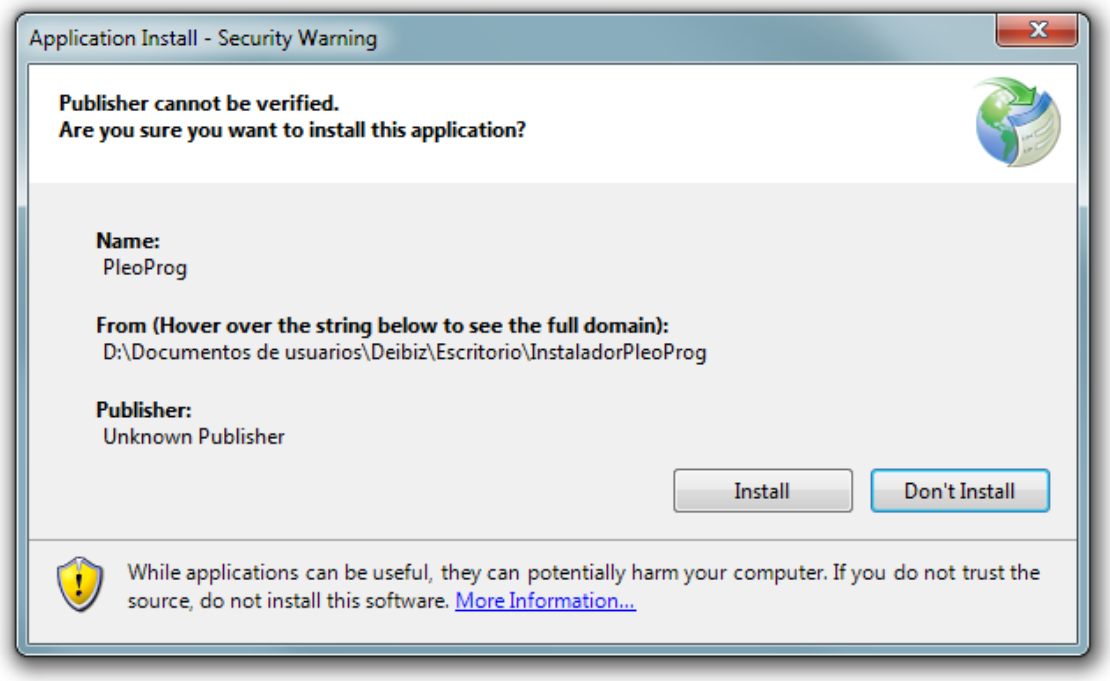


Figura 9.2 Advertencia de instalación

Al finalizar la instalación la aplicación se abrirá de manera automática. A partir de este momento se podrá ejecutar PleoProg siempre que desee. Busque en su barra de inicio, en “Todos los programas” “DavidGG”, dentro encontrará una carpeta con el programa PleoProg.

9.2 Manual de Usuario

A través del siguiente manual podrá aprender el funcionamiento de la herramienta y como programar para Pleo, actualmente la única forma de vida para la que tiene soporte.

9.2.1 Primeros Pasos

Al abrir la herramienta se mostrará un cuadro como el siguiente:

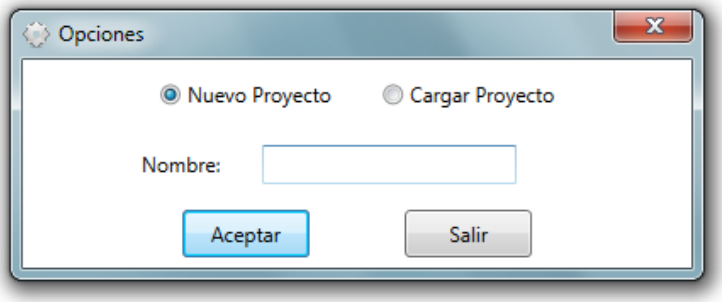


Figura 9.3 Cuadro inicio

Puede elegir entre crear un nuevo proyecto, escribiendo el nombre deseado, o cargar un proyecto previamente guardado, indicando la ruta de su ubicación. Si es la primera vez que ejecuta la herramienta haga clic en Nuevo Proyecto y escriba un nombre que desee dar.

Se encontrará ante la pantalla principal:

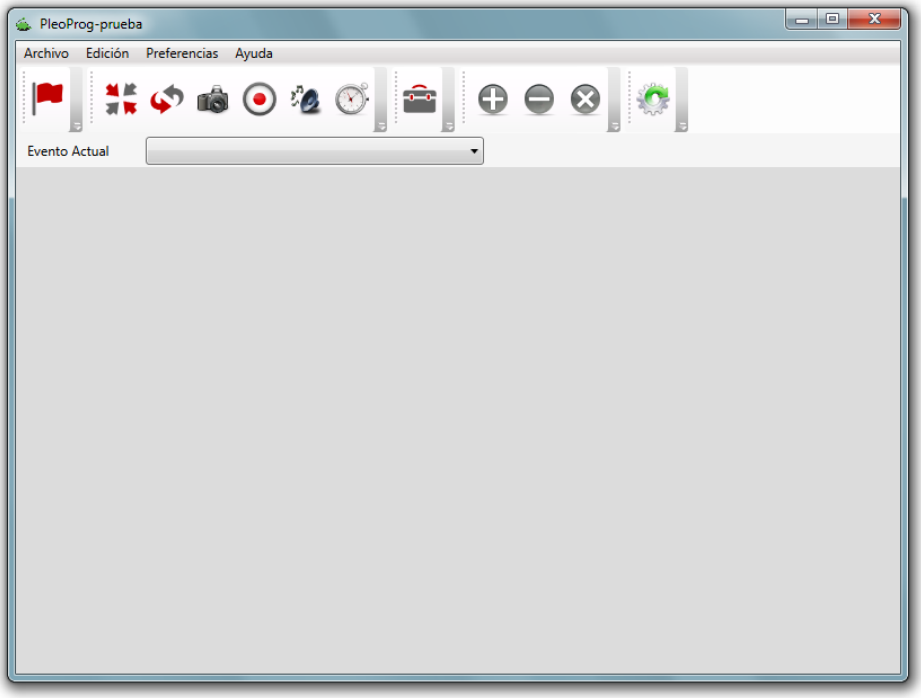


Figura 9.4 Pantalla principal

En la parte superior encontrará un menú con las opciones **Archivo**, **Edición**, **Preferencias** y **Ayuda**. Cada una de estas opciones viene descrita en las siguientes secciones. Debajo de esta barra se encontrará con una nueva barra de herramientas, los botones que se encuentran en la primera y segunda barra de herramientas son botones interruptores. Esto quiere decir que mientras esté activo podrá crear todas las órdenes de un mismo tipo que desee sin necesidad de volver a pulsarlo. Esta barra de herramientas contiene las opciones necesarias para crear una personalidad. En primer lugar se encuentra el botón de **Nuevo Evento**. A continuación 6 botones de órdenes que representan **Posición Normal**, **Movimiento**, **Captura de Imagen**, **Captura de Sonido**, **Reproducción de Sonido** y **Tiempo en Espera**. El siguiente botón permite incluir una **Animación**. Los 3 siguientes botones sirven para **Aumentar** y **Disminuir el Turno** y para **Eliminar una Orden o Evento**. Por último se encuentra el botón de **Compilado** que inicia el compilado de la personalidad creada.

Debajo de la botonera principal se encuentra un campo que mostrará el **Evento Actual** y que permite seleccionar cualquiera de los que se hayan añadido. La utilidad de este cuadro es dar rápidamente información sobre que evento es el activo y sobre cual se añadirá la siguiente orden que se introduzca.

Antes de seguir se recomienda que el primer paso sea configurar las preferencias de PleoProg. Para ello será necesario que descargue en primer lugar el software PDK que permite la compilación de las personalidades creadas, puede descargarlo desde aquí: http://www.pleoworld.com/downloads/pdk.aspx

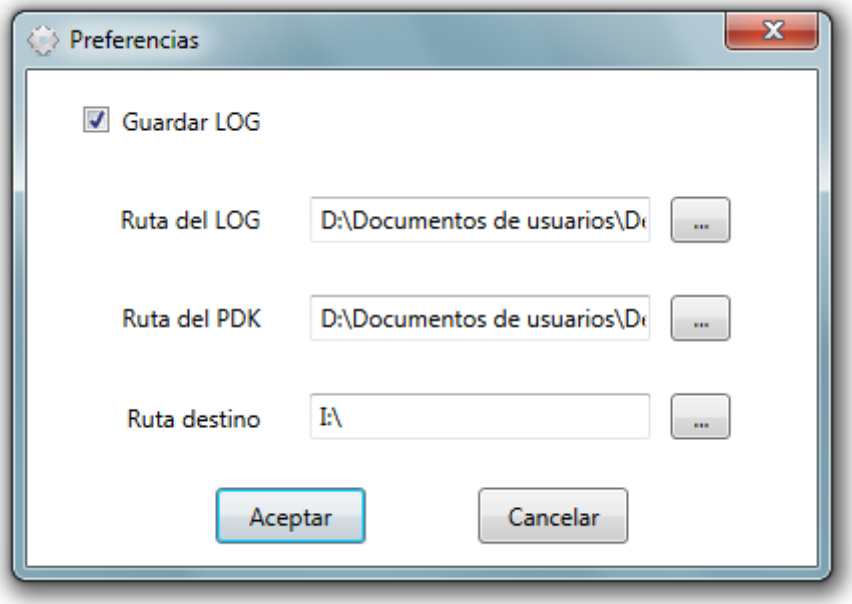


Figura 9.5 Cuadro de preferencias`

Seleccione si desea guardar un LOG, la ruta del LOG y la ruta de la carpeta que ha descargado de la página nombrada más arriba. Seleccione una ruta por defecto donde guardar el archivo resultante de la compilación, para su comodidad lo mejor sería elegir la ruta de la tarjeta SD que va a insertar en la forma de vida artificial.

9.2.2 Crear una Personalidad

9.2.2.1 Eventos

Los eventos son los estímulos exteriores que bajo ciertas circunstancias Pleo detecta. Para crear una personalidad es necesario agregar las órdenes o reacciones de Pleo a estos eventos.

Abra un nuevo proyecto y elija un nombre. Una vez se encuentre con la ventana principal de la herramienta active el botón **Nuevo Evento** y pulse dentro del cuadro de trabajo, donde desee crear dicho evento. Se abrirá una ventana en la que podrá seleccionar el tipo de evento que desea añadir.

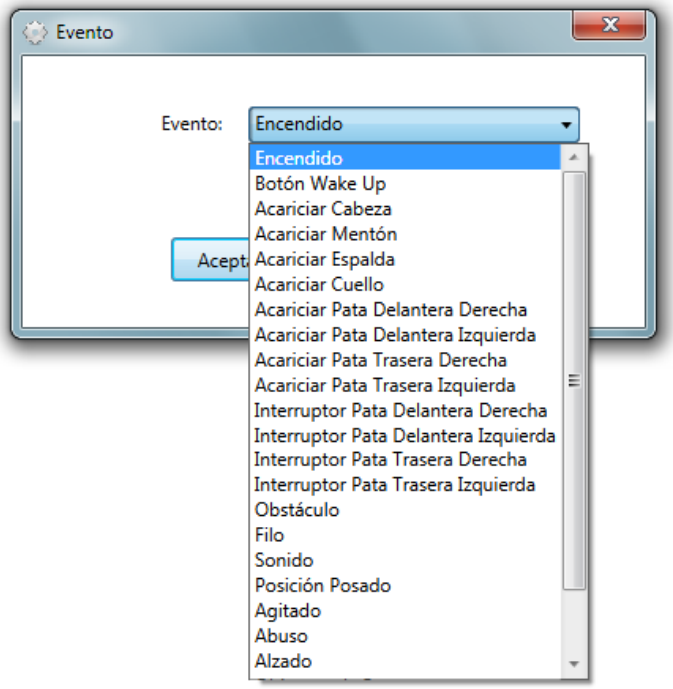


Figura 9.6 Cuadro inserción evento

Los diferentes tipos de eventos que puede seleccionar son:

- **Encendido:** Evento que se dispara cuando se enciende.
- **Botón Wake Up:** Evento que se dispara cuando se pulsa el botón situado cerca de la ranura de la tarjeta SD.
- **Acariciar Cabeza:** Evento que se dispara cuando se acaricia la parte superior de la cabeza.
- **Acariciar Mentón:** Evento que se dispara cuando se acaricia la parten inferior de la cabeza, o mentón.
- **Acariciar Espalda:** Evento que se dispara cuando se acaricia la parte inferior del tronco en la espalda.

- **Acariciar Cuello:** Evento que se dispara cuando se acaricia la parte suprior del tronco en la espalda, o cuello.
- **Acariciar Pata Delantera Derecha:** Evento que se dispara cuando se acaricia la pata delantera derecha.
- **Acariciar Pata Delantera Izquierda:** Evento que se dispara cuando se acaricia la pata delantera izquierda.
- **Acariciar Pata Trasera Derecha:** Evento que se dispara cuando se acaricia la pata trasera derecha.
- **Acariciar Pata Trasera Izquierda:** Evento que se dispara cuando se acaricia la pata trasera izquierda.
- **Interruptor Pata Delantera Derecha:** Evento que se dispara cuando se pulsa el interruptor de pie de la pata delantera derecha.
- **Interruptor Pata Delantera Izquierda:** Evento que se dispara cuando se pulsa el interruptor de pie de la pata delantera izquierda.
- **Interruptor Pata Trasera Derecha:** Evento que se dispara cuando se pulsa el interruptor de pie de la pata trasera derecha.
- **Interruptor Pata Trasera Izquierda:** Evento que se dispara cuando se pulsa el interruptor de pie de la pata trasera izquierda.
- **Obstáculo:** Evento que se dispara cuando se detecta un obstáculo. Depende de la inclinación del cuello para poder ser disparada.
- **Filo:** Evento que se dispara cuando se detecta un filo. Depende de la inclinación del cuello para poder ser disparada.
- **Sonido:** Evento que se dispara cuando se detecta un sonido.
- **Posición Posado:** Evento que se dispara cuando se detecta que se ha posado.
- **Agitado:** Evento que se dispara cuando es agitado.
- **Abuso:** Evento que se dispara cuando se intenta mover las extremidades.
- **Alzado:** Evento que se dispara cuando se detecta que está alzado.
- **Objeto en la Boca:** Evento que se dispara cuando se detecta un objeto en la boca.
- **Variación de Luz:** Evento que se dispara cuando se detecta una variación de luz.
- **Otro Pleo:** Evento que se dispara cuando se detecta otro Pleo.

A la hora de añadir un nuevo evento debe tener presente que no puede agregar un evento repetido.

9.2.2.2 Órdenes

Las órdenes son las acciones que llevará a cabo Pleo. Es imprescindible que se haya añadido un evento antes de intentar insertar una orden.

Para insertar una orden elija en la botonera entre las órdenes disponibles, active el botón y haga clic en el entorno de trabajo donde haya un hueco libre y desee añadir la orden. Tenga en cuenta el evento actual, que se mostrará, debajo de la barra de herramientas para saber a qué evento se añadirá dicha orden.

Las órdenes disponibles están divididas en los siguientes tipos:

- **Posición Neutra:** Mueve la forma de vida artificial a la posición neutra, mueve todos los miembros a su posición normal.
- **Movimiento:** Mueve un miembro a una posición determinada. Este movimiento puede ser **Único** o **Repetitivo**
- **Captura de Imagen:** Captura una imagen y la guarda en la tarjeta SD con el nombre que el usuario especifique.
- **Captura de Sonido:** Captura sonido y lo guarda en la tarjeta SD con el nombre y duración que el usuario especifique.
- **Reproducción de Sonido:** Reproduce una pista de sonido que esté en formato .WAV. Es importante tener en cuenta que por motivos de ahorro de memoria no se permiten archivos de más de 1MB de tamaño.
- **Tiempo en Espera:** Establece un periodo de espera en el que no se realizará ninguna orden.
- **Animaciones:** Reproduce una animación que esté en formato .CSV. Es importante tener en cuenta que por motivos de ahorro de memoria no se permiten archivos de más de 1MB de tamaño.

Cuando la orden lo requiera se mostrará una ventana donde el usuario deberá insertar los datos necesarios para llevar a cabo la orden. A continuación se detallan los parámetros necesarios para algunas órdenes.

9.2.2.2.1 Movimiento

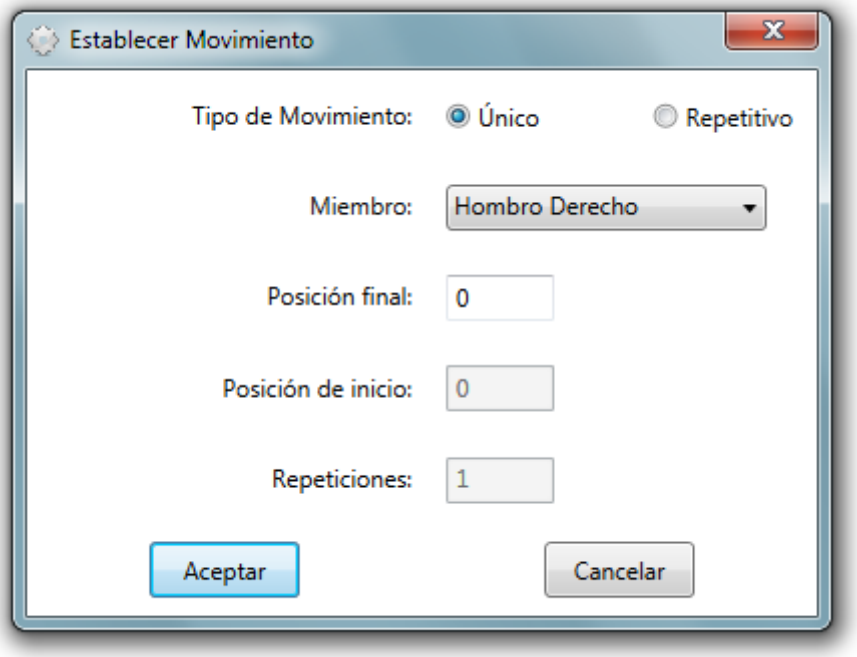


Figura 9.7 Cuadro inserción orden movimiento

Al insertar una orden de movimiento podrá elegir si desea que sea un movimiento único o repetitivo. Para ambos casos debe completar los siguientes campos:

- **Miembro**: Miembro de la forma de vida a mover.
- **Posición final**: Posición final del miembro al terminar el movimiento, en grados.

Si además se trata de un movimiento repetitivo deberá completar:

- **Posición de inicio**: Posición de inicio del movimiento, en grados.
- **Repeticiones**: Número de veces que se realizará el movimiento.

Tenga en cuenta que los grados que introduzca tienen que estar dentro de los parámetros que soporte el hardware de la forma de vida.

9.2.2.2.2 Captura de imagen

En el momento de introducir en una secuencia la orden captura de una imagen verá un cuadro como el siguiente donde debe introducir el nombre con el que se guardará el archivo en la tarjeta SD.

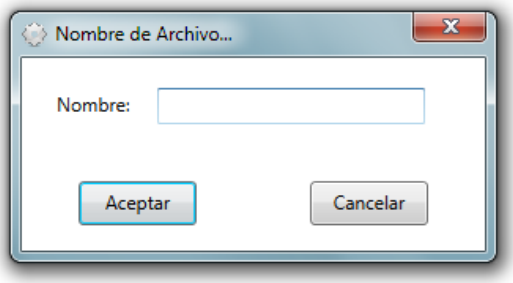


Figura 9.8 Cuadro captura imagen

9.2.2.2.3 Captura de sonido

En el momento de introducir en una secuencia la orden de captura de sonido verá un cuadro como el siguiente donde debe introducir el nombre con el que se guardará el archivo en la tarjeta SD y el tiempo de grabación.

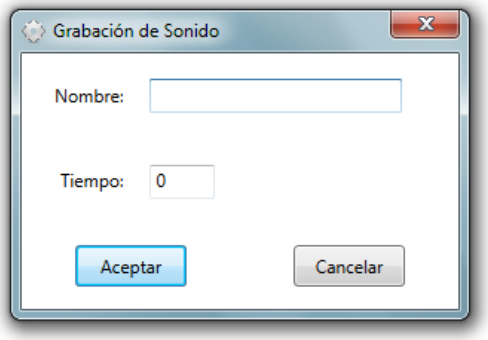


Figura 9.9 Cuadro captura sonido

9.2.2.2.4 Tiempo en espera

Al elegir una orden de tiempo en espera deberá introducir los segundos de espera.

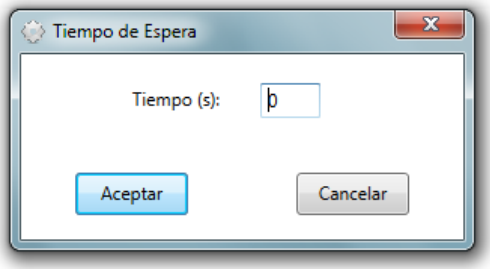


Figura 9.10 Cuadro de orden de tiempo en espera

9.2.2.2.5 Reproducir Sonido y Animación

En el caso de estas dos órdenes se abrirá una ventana del explorar de archivos y deberá seleccionar el archivo .WAV (reproducir sonido) o .CSV (animación) que desea adjuntar.

9.2.2.2.6 Posición Neutra

La orden de posición neutra no requiere de ningún parámetro por lo que al hacer clic sobre el área de trabajo cuando el botón correspondiente esta activo, añadirá dicha orden sin ningún tipo de inserción de datos por parte del usuario.

9.2.2.3 Compilar

La compilación es el paso final para obtener un archivo que pueda ejecutar la forma de vida artificial elegida, actualmente solo Pleo.

Para llevar a cabo la compilación debe crear un diagrama que represente una personalidad válida utilizando los eventos y órdenes. Una vez tenga la personalidad creada deberá pulsar el último botón de la botonera **Compilar**. Se abrirá una ventana como la siguiente donde podrá configurar algunas opciones antes del proceso de compilado.

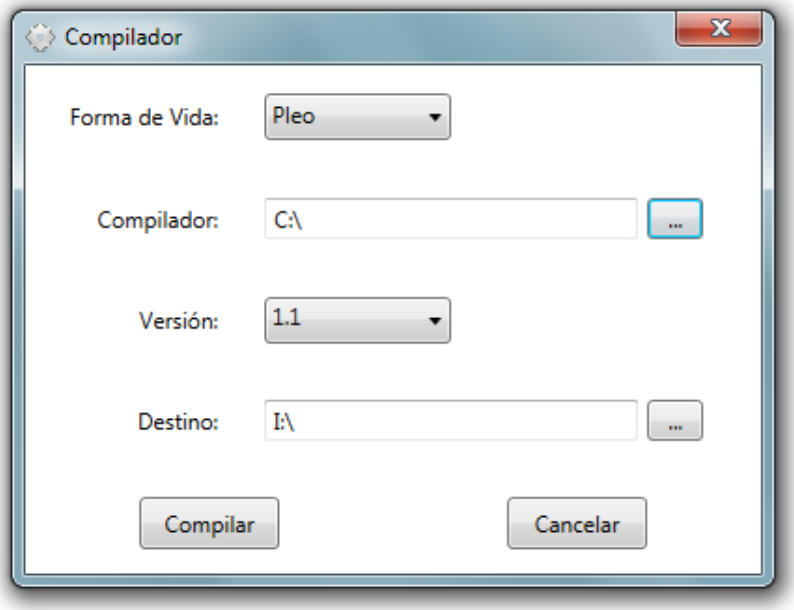


Figura 9.11 Cuadro de compilación

Elija la forma de vida y podrá seleccionar la versión de dicha forma de vida, si procede. Elija la ruta del compilador y la ruta de destino del archivo ejecutable. Puede seleccionar la ubicación de la tarjeta SD que vaya a utilizar o cualquier otra ruta.

Al pulsar sobre el botón compilar verá un mensaje con el resultado del compilado.

9.2.3 Nueva Personalidad

Al arrancar la aplicación o en cualquier otro momento es posible crear una nueva personalidad. Al arrancar se mostrará un diálogo como el siguiente donde podrá elegir si desea crear una nueva personalidad o cargar una guardada previamente.

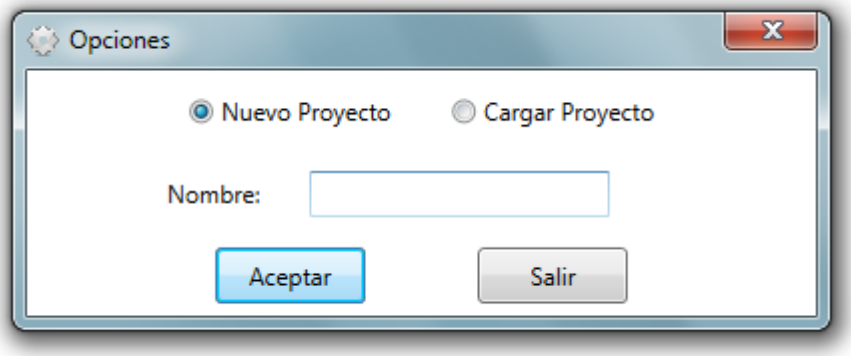


Figura 9.12 Cuadro de inicio

Escriba un nombre de proyecto y pulse en aceptar.

En cualquier otro momento puede ir al menú **Archivo** y pulsar sobre **Nuevo...**. La herramienta le preguntará si desea guardar la personalidad actual y posteriormente le mostrará un cuadro donde introducir el nombre del nuevo proyecto.

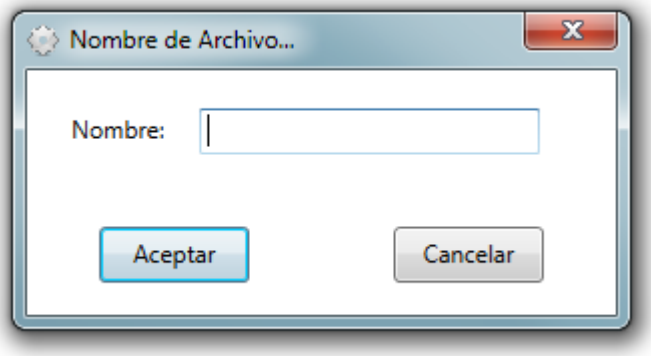


Figura 9.13 Cuadro nueva personalidad

9.2.4 Guardar Personalidad

Para guardar la personalidad actual, que está editando, debe ir al menú Archivo y pulsar sobre Guardar... La herramienta le mostrará un cuadro para que elija el lugar y el nombre del archivo .BIN que se guardará con los datos de su proyecto.

9.2.5 Cargar Personalidad

Al arrancar la aplicación o en cualquier otro momento es posible cargar una personalidad creada previamente. Al arrancar se mostrará un diálogo como el siguiente donde podrá elegir si desea crear una nueva personalidad o cargar una guardada previamente.

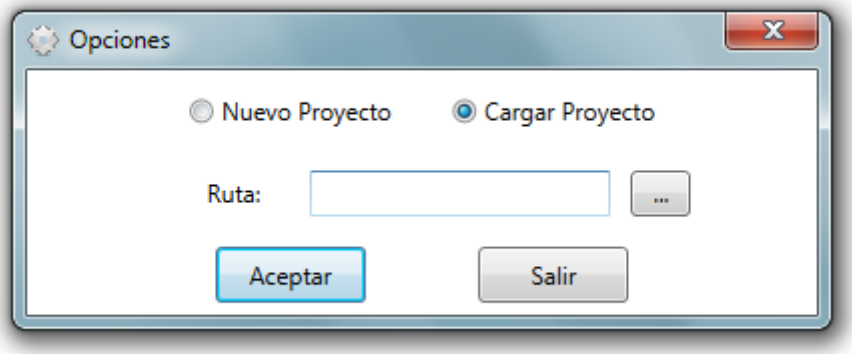


Figura 9.14 Cuadro cargar proyecto

Seleccione la ruta del proyecto (archivo .BIN) y pulse en aceptar.

En cualquier otro momento puede ir al menú **Archivo** y pulsar sobre **Cargar...**. La herramienta le preguntará si desea guardar la personalidad actual y posteriormente le mostrará un diálogo para que seleccione el archivo a cargar.

Después de cargar el archivo la herramienta colocará todos los elementos de una forma sencilla, sea cual fuere el anterior aspecto del diagrama.

9.2.6 Modificación de eventos y órdenes

El proceso para modificar un evento u orden es igual en ambos casos, tan sólo es necesario hacer doble clic sobre el evento u orden a modificar y la herramienta mostrará un diálogo con las posibles opciones de modificación.

En el caso de los eventos se mostrará una ventana con todos los eventos disponibles para que se pueda seleccionar el nuevo evento que sustituirá al antiguo. Este nuevo evento no deberá estar añadido en el diagrama pues cada evento sólo puede aparecer una única vez.

En el caso de las órdenes se mostrará una ventana con los campos modificables de la orden. Esta ventana será diferente para cada orden. Tenga en cuenta que la orden Posición Normal no tiene ningún campo modificable y tan sólo se podrá eliminar.

Es posible la eliminación de los eventos y órdenes. Para ello debe seleccionar el evento u orden a eliminar y pulsar sobre el menú **Edición** y luego sobre **Eliminar** o directamente usar el botón habilitado para tal función, situado en la penúltima barra, anterior al botón de **Compilar**

Los eventos que posean alguna orden mostrarán el siguiente cuadro de información al intentar borrarlos.

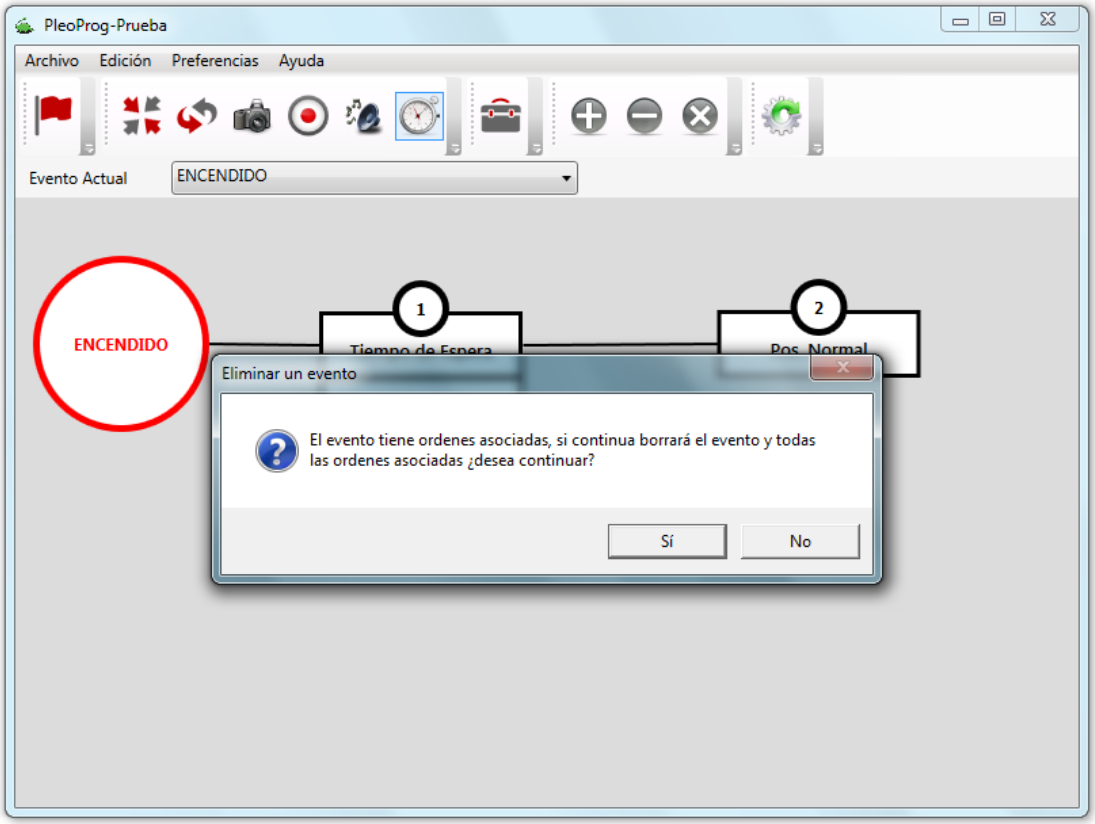


Figura 9.15 Cuadro eliminación de evento con órdenes

El usuario podrá decidir si cancelar la operación o borrar en cascada dicho evento con todas sus órdenes.

9.2.7 Modificación del turno de las órdenes

Las órdenes se ejecutan una vez que el evento que la lanza es detectado por Pleo, estas órdenes se ejecutan una a una siguiente un orden o turno. El turno es designado según se van añadiendo las órdenes. Sin embargo es posible modificar el turno de la orden para que se ejecute antes o después de su turno actual.

Para modificar el turno hay dos posibilidades **Aumentar el Turno** o **Disminuir el Turno**. Es necesario estar editando una personalidad y haber seleccionado la orden a la que se va a modificar el turno. Una vez seleccionada se puede aumentar o disminuir el turno usando los botones a tal efecto o a través del menú **Edición** y **Aumentar el Turno** o **Disminuir el Turno**. Los únicos requisitos para aumentar el turno o disminuirlo es que a la orden le suceda otra orden (aumentar turno) o que a la orden le preceda otra orden (disminuir turno). Esta modificación de turno se realiza intercambiando la posición con el anterior o posterior.

En el siguiente ejemplo se puede observar como el siguiente diagrama se modifica cuando a la orden **Tiempo de Espera** se le disminuye el turno.

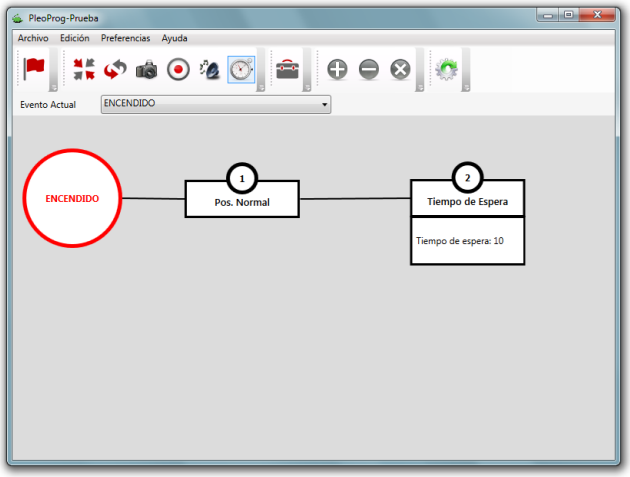


Figura 9.16 Antes de modificar el turno

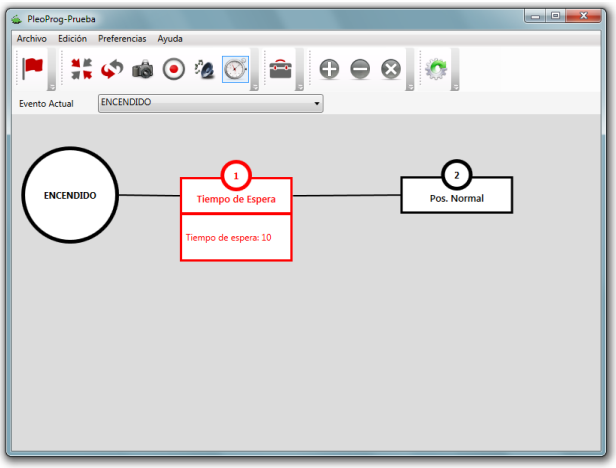


Figura 9.17 Después de modificar el turno

9.3 Manual del Programador

9.3.1 Añadir soporte para una nueva versión de Pleo

Actualmente PleoProg presta soporte para las versiones de Pleo 1.0 y 1.1. Dar soporte para una nueva versión de Pleo es una tarea no demasiado compleja, ya que tan solo es necesario modificar una única clase y añadir una nueva versión disponible al comboBox mostrado en la ventana WindowCompilador y ubicada en el paquete Interfaz.

Para dar soporte a una nueva versión abra el archivo "CompiladorPleo.cs" dentro del paquete Compilador del código fuente. Dependiendo de los nuevos cambios deberá realizar cambios en los métodos que crean cada uno de los archivos de entrada. Utilice la variable iVFormaVida para saber que versión de la forma de vida ha elegido el usuario, realizando los cambios pertinentes en función de esta variable.

9.3.2 Añadir soporte para una forma de vida

La herramienta se ha construido pensando en la posibilidad de ampliar los compiladores. Debe tener en cuenta los siguientes aspectos a la hora de añadir una nueva forma de vida.

La clase Util posee variables para identificar miembros, movimientos, eventos y órdenes, es posible que deba modificar esta clase si quiere dar soporte completo para la nueva forma de vida artificial.

Añada más clases de tipo Orden si fuera necesario, o modifique las existentes para que permitan ciertos parámetros que la nueva forma de vida acepta y no las anteriores.

Añada un nuevo compilador que herede de la clase Compilador. Esta es la modificación más importante, y en ciertos casos, sería la única. La herramienta genera un archivo XML con la personalidad, tan solo debe crear una nueva clase compilador que interprete este archivo creando los archivos necesarios para la nueva forma de vida.

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

La herramienta desarrollada cumple con todos los objetivos previstos en las primeras fases del proyecto.

Se ha desarrollado una herramienta sencilla, intuitiva y eficaz para programar formas de vida artificiales. Se trata pues de una aplicación que puede ser utilizada por cualquier persona, con conocimientos o no de informática. Para llegar a esto se ha diseñado una interfaz lo más natural y sencilla posible que utiliza los iconos y símbolos para que sean más fáciles de identificar las órdenes.

En cuanto a la tecnología utilizada, WPF, es sin duda un campo a tener muy en cuenta. Propone, dentro del desarrollo de aplicaciones, potenciar el trabajo en grupo facilitando a través de diferentes herramientas (Microsoft Blend, Visual Studio, etc.) el trabajo de diferentes profesionales a través de diferentes flujos. Personalmente me ha gustado mucho dicha tecnología.

El lenguaje de programación, muy similar a Java, ha demostrado su potencia a lo largo del desarrollo del proyecto.

10.2 Ampliaciones

Inicialmente estas formas de vida se encuentran limitadas a Pleo, en sus versiones 1.0 y 1.1 pero que en el futuro pueden ser ampliadas a nuevas versiones u otras formas de vida. Se ha contemplado dicha posibilidad de cara a que la herramienta pueda ser utilizada como una única plataforma de programación de varias formas de vida. No se ha llevado a cabo la implementación de otras formas de vida debido a la complejidad y al incremento temporal que significaría, pues sería necesaria más investigación para ello.

Otras posibles ampliaciones son las traducciones a otros idiomas, por falta de tiempo no se han implementando más idiomas, pero en un plazo corto de tiempo se creará una versión en inglés.

Además de estas ampliaciones también, aprovechando la relación entre WPF y Silverlight sería posible crear una aplicación web con Silverlight para que cualquier usuario tenga acceso a la herramienta vía web. Para ello habría que analizar ciertas partes de código que se debería reescribir para Silverlight por no tener algunas librerías.

Capítulo 11. Presupuesto

11.1 Presupuesto de Costes

A continuación se detalla el presupuesto de costes del proyecto realizado:

Ítem	Subítem	Concepto	Horas	Precio Unitario	Total
0		**Investigación**			**7.200,00 €**
	1	Aspectos Generales	60	30,00 €	1.800,00 €
	2	PLEO	30	30,00 €	900,00 €
	3	Tecnologias Utilizar	150	30,00 €	4.500,00 €
1		**Análisis**			**2.320,00 €**
		Análisis	58	40,00 €	2.320,00 €
2		**Diseño**			**1.120,00 €**
	1	Arquitectura	14	40,00 €	560,00 €
	2	Interfaz	14	40,00 €	560,00 €
3		**Implementación**			**1.560,00 €**
	1	Implementación	78	20,00 €	1.560,00 €
4		**Pruebas**			**400,00 €**
	1	Pruebas	20	20,00 €	400,00 €
Ítem		**Concepto**	**Unidades**	**Precio Unitario**	**Total**
5		**Material adicional**			**650,00 €**
	1	Pleo	2	300,00 €	600,00 €
	2	Bateria extra	1	50,00 €	50,00 €
Ítem		**Concepto**	**Tiempo (meses)**	**Precio Unitario**	**Total**
6		**Otros gastos**			**900,00 €**
	1	Oficina	9	100,00 €	900,00 €
				Subtotal	14.150,00 €
				IVA (18%)	2.547,00 €
				TOTAL	16.697,00 €

Figura 11.1 Presupuesto de costes detallado

Este presupuesto es un presupuesto de la organización que sirve para determinar los costes de desarrollo del proyecto. A partir de este presupuesto se elabora un presupuesto para el cliente en el que se incluye un margen para el beneficio. Para desarrollar este presupuesto se calcula para los conceptos que se presentarán en el presupuesto del cliente el coste y se aplicará un beneficio de un 20%.

Los conceptos que se presentarán al cliente serán los ítems del anterior presupuesto, excepto el ítem 5 que será incluido dentro de la implementación y el ítem 6 que será incluido entre todos los ítems. Estos dos ítems no se encuentran detallados en el presupuesto del cliente por no creerse conveniente que se le muestren.

A continuación una relación de los costes de cada uno de estos ítems (incluyendo los ítems 5 y 6 ya dentro de estos) y el resultado de aplicarle el beneficio. El siguiente presupuesto se muestra para ilustrar el proceso de cálculo de presupuesto para el cliente, pero su destino no es el cliente final.

Ítem	Concepto	Coste	Otros ítems	Subtotal	Beneficio(20%)
0	**Investigación**	7.200,00 €	180,00 €	**7.380,00 €**	**8.856,00 €**
1	**Análisis**	2.320,00 €	180,00 €	**2.500,00 €**	**3.000,00 €**
2	**Diseño**	1.120,00 €	180,00 €	**1.300,00 €**	**1.560,00 €**
3	**Implementación**	1.560,00 €	830,00 €	**2.390,00 €**	**2.868,00 €**
4	**Pruebas**	400,00 €	180,00 €	**580,00 €**	**696,00 €**
				Subtotal	16.980,00 €
				IVA (18%)	3.056,40 €
				TOTAL	20.036,40 €

Figura 11.2 Cuadro calculo de beneficio

11.2 Presupuesto del cliente

Ítem	Concepto	Total
0	**Investigación**	8.856,00 €
1	**Análisis**	3.000,00 €
2	**Diseño**	1.560,00 €
3	**Implementación**	2.868,00 €
4	**Pruebas**	696,00 €
	Subtotal	16.980,00 €
	IVA (18%)	3.056,40 €
	TOTAL	20.036,40 €

Figura 11.3 Presupuesto del cliente

La **investigación** es todo el proceso previo de recogida de datos de las tecnologías a utilizar, el campo a desarrollar, los materiales implicados (en este caso la forma de vida Pleo), alternativas etc.

El **análisis** describe qué se desarrollará. Después de su realización se concretará una reunión para su aprobado.

El **diseño** del proyecto incluyo cómo se desarrollará el proyecto y que decisiones se tomarán para probarlo. Al igual que en el análisis se concretará una reunión para aprobarlo.

La **implementación** incluye todo el proceso de creación de la herramienta a elaborar.

Las **pruebas** son la parte que prueba el sistema elaborado de múltiples formas.

Capítulo 12. Referencias Bibliográficas

12.1 Libros y Artículos

Freudenthal, M, y Cybernetica AS, Tallinn. 2009. «Domain Specific Languages in a Customs Information System». *Software, IEEE* PP Issue: 99: 1.

Leffingwell, Dean. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series)*.

Wilson, Robert Andrew, y Frank C. Keil. *The MIT encyclopedia of the cognitive science*.

Sergi Bermejo. 2003. *Desarrollo de robots basados en el comportamiento*.

Mark A. Bedau. 2003. «Artificial life: organization, adaptation and complexity from the bottom up». *Elsevier, TRENDS in Cognitive Sciences* Vol.7 No.11 http://people.reed.edu/~mab/publications/papers/BedauTICS03.pdf (Accedido Mayo 7, 2011).

12.2 Referencias en Internet

AiboHack. 2008. « [AiboHack] YAPT = Yet Another Pleo Tool (1.0 and 1.1 compatible with Babe)». http://www.aibohack.com/pleo/yapt.htm (Accedido Septiembre 29, 2010).

Amit. 2009. «How to Create a Drag & Drop / Move, Zoom In & Out Content Control | Dev102.com». http://www.dev102.com/2009/02/23/how-to-create-a-drag-drop-move-zoom-in-out-content-control/ (Accedido Junio 1, 2011).

Anderson, Tim. 2004. «Introducing XML, its history, what it is, its significance». http://www.itwriting.com/xmlintro.php (Accedido Mayo 7, 2011).

asimo. «ASIMO's Specifications». http://asimo.honda.com/asimo_specifications.html (Accedido Septiembre 29, 2010).

Basaric, Denis. 2009. «WPF Tip: Label Word Wrap». http://devcomponents.com/blog/?p=484 (Accedido Junio 7, 2011).

Borel, Brooke. 2010. «A Ping-Pong-Playing Terminator | Popular Science». http://www.popsci.com/technology/article/2010-02/ping-pong-playing-terminator (Accedido Septiembre 29, 2010).

Bray, Tim. 1996. «Extensible Markup Language (XML)». http://www.w3.org/TR/WD-xml-961114.html (Accedido Mayo 7, 2011).

«C# Language Specification at C# Online.NET». http://en.csharp-online.net/CSharp_Language_Specification (Accedido Mayo 7, 2011).

C. M. Sperberg-McQueen. 1997. «Reports from the SGML ERB to the SGML WG and from the XML WG to the XML SIG». http://www.w3.org/XML/9712-reports.html (Accedido Mayo 7, 2011).

Cowan, John, y Richard Tobin. 2004. «Conjunto de Información XML (Segunda edición)». http://www.spanish-translator-services.com/espanol/t/infoset.htm (Accedido Mayo 7, 2011).

dotnetero. 2009. «Algo de Historia de C#». http://dotnetero.blogspot.es/ (Accedido Mayo 7, 2011).

electronicpets. «Sony Aibo ERS-7 at ElectronicPets.org». http://www.electronicpets.org/sony-aibo-ers7~p14.html (Accedido Septiembre 29, 2010).

Endres, Christoph, y Daniel Braun. «PleopatraTools». http://www.dfki.de/pleopatra/ (Accedido Octubre 13, 2011).

Engber, Daniel. 2008. «A shopper's guide to robot pets. - By Daniel Engber - Slate Magazine». http://www.slate.com/id/2206604/?y= (Accedido Septiembre 29, 2010).

Feinbube, Frank. 2009. «PDK (pleo development kit)». http://bobthepleo.com/forums/index.php?topic=360.0 (Accedido Febrero 20, 2011)

Fowler, Martin. «DomainSpecificLanguage». http://www.martinfowler.com/bliki/DomainSpecificLanguage.html (Accedido Septiembre 23, 20109

Genuth, Iddo. 2006. «The Rise and Fall of ASIMO». http://thefutureofthings.com/pod/121/the-rise-and-fall-of-asimo.html (Accedido Septiembre 29, 2010).

GIMP. «Descargas, tutoriales, noticias e información de GIMP | Alternativa libre a Photoshop | - GIMP-Es». http://www.gimp.org.es/ (Accedido Mayo 1, 2011).

GIMP. «GIMP - The GNU Image Manipulation Program». http://www.gimp.org/ (Accedido Mayo 1, 2011).

Hara, Yoshiko. 2000. «Latest robots fill helper, entertainer roles». http://www.eetimes.com/electronics-news/4168362/Latest-robots-fill-helper-entertainer-roles (Accedido Septiembre 29, 2010).

HONDA. «Honda Worldwide | ASIMO». http://world.honda.com/ASIMO/ (Septiembre 29, 2010).

Kertesz, Csaba. 2009. «AiBO+». http://aiboplus.sourceforge.net/ (Accedido Septiembre 29, 2010).

logoinstant. «Dinosauroz | Logo Instant». http://www.logoinstant.com/2009/06/dinosauroz/# (Accedido Mayo 1, 2011).

«Metrica v3 - Portal de Administración electrónica». http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=P60085901274201580632&langPae=es (Accedido Mayo 20, 2011).

Morales Díaz, Leonel. 2009. «Jef Raskin y las Leyes del Diseño de Interfaces « Ingeniería Simple». http://ingenieriasimple.com/blog/?p=125 (Accedido Mayo 1, 2011).

Moser, Christian. 2011. «WPF Tutorial». http://www.wpftutorial.net/ (Accedido Febrero 7, 2011).

msdn. «Application Settings». http://msdn.microsoft.com/en-us/library/a65txexh.aspx (Accedido Abril 18, 2011).

msdn. «C# Programming Guide». http://msdn.microsoft.com/en-us/library/67ef8sbd.aspx (Accedido Noviembre 12, 2010).

msdn. 2010. «Cómo: Crear archivos o carpetas (Guía de programación de C#)». http://msdn.microsoft.com/es-es/library/as2f1fez.aspx (Accedido Mayo 7, 2011).

msdn «Directory.Delete Method (String, Boolean) (System.IO)». http://msdn.microsoft.com/en-us/library/fxeahc5f.aspx (Accedido Mayo 7, 2011).

msdn. «File and Stream I/O». http://msdn.microsoft.com/en-us/library/k3352a4t.aspx (Accedido Mayo 7, 2011).

msdn. «File.Copy Method (String, String, Boolean) (System.IO)». http://msdn.microsoft.com/en-us/library/9706cfs5.aspx (Accedido Mayo 7, 2011).

msdn. «How to: Add or Remove Application Settings». http://msdn.microsoft.com/en-us/library/25zf0ze8.aspx (Accedido Abril 18, 2011).

msdn. «How to: Create a Directory Listing». http://msdn.microsoft.com/en-us/library/5cf8zcfh.aspx (Accedido Noviembre 24, 2010).

msdn. «Introducing Windows Presentation Foundation». http://msdn.microsoft.com/en-us/library/aa663364.aspx#introducingwpf_topic2 (Accedido Noviembre 14, 2010).

msdn. «MSDN Blogs». http://blogs.msdn.com/b/pedrosilva/ (Accedido Noviembre 14, 2010).

msdn. «The C# Language». http://msdn.microsoft.com/en-us/vcsharp/aa336809.aspx (Accedido Noviembre 14, 2011).

msdn. «XAML Overview (WPF)». http://msdn.microsoft.com/en-us/library/ms752059.aspx (Accedido Noviembre 14, 2011).

MySkit. «MySkit - Performance Editor for PLEO». http://www.dogsbodynet.com/myskit/index.html (Accedido Febrero 20, 2011).

«Nuevo dinosaurio robótico califica como una forma de vida artificial». 2000. http://www.compute-rs.com/es/consejos-480214.htm (Accedido Febrero 20, 2011).

Quin, Liam. 2011. «Extensible Markup Language (XML)». http://www.w3.org/XML/ (Accedido Enero 14, 2011).

RAE. «Robótica, Diccionario de la lengua española - Vigésima segunda edición». http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=rob%C3%B3tica (Accedido Junio 1, 2011).

sony. «Sony Global - Product & Technology Milestones-Robot». http://www.sony.net/SonyInfo/CorporateInfo/History/sonyhistory-j.html (Accedido Octubre 18, 2010).

Spinellis, Diomidis. «Notable Design Patterns for Domain-Specific Languages». http://www.spinellis.gr/pubs/jrnl/2000-JSS-DSLPatterns/html/dslpat.html (Accedido Septiembre 23, 2010).

sukram. «WPF Diagram Designer: Part 1 - CodeProject». http://www.codeproject.com/KB/WPF/WPFDiagramDesigner_Part1.aspx (Accedido Junio 1, 2011).

tosy. «TOPIO - TOSY Ping-Pong Playing Robot». http://topio.tosy.com/about.shtml (Accedido Septiembre 13, 2010).

Ugobe. 2007. «Download Different "Pleo Personalities"». http://www.pleoworld.com/downloads/personalitydownloads.aspx (Accedido Febrero 20, 2011).

PleoWorld. 2008. «Pleo Developer Resources». http://www.pleoworld.com/developers.aspx (Accedido Febrero 20, 2011).

PleoWorld. «Tools for Pleo | Make Pleo Dance and Do Skits!» http://www.pleoworld.com/downloads/tools.aspx (Accedido Febrero 20, 2011).

Williams, Martyn. 2006. «Sony reports strong earnings but unplugs Aibo | Default | Macworld». http://www.macworld.com/article/49118/2006/01/sony.html (Accedido Enero 10, 2011).

WPFDev. 2010. «WPF and Silverlight design and development | WPF Dev». http://www.wpfdev.com/ (Accedido Enero 20, 2011).

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

Por orden alfabético, todos los términos que se consideren importantes en la aplicación con una descripción breve de su significado dentro de la aplicación.

- **.NET Framework:** Framework de Microsoft que permite que una aplicación desarrollada para esta plataforma pueda ser ejecuta en diferentes sistemas operativos que la tenga instalada.
- **Abuso:** Evento que se lanza cuando se intenta mover externamente alguna de las partes de la forma de vida.
- **Agitado:** Evento que se lanza cuando se agita a la forma de vida.
- **Alzado:** Evento que se lanza cuando se levanta la forma de vida.
- **Animación o Motion:** Se trata de un conjunto de movimientos almacenados en un mismo archivo y que se ejecutan con un ritmo predefinido. En el presente contexto estos archivos están guardados con extensión.csv
- **Batch:** Archivo de procesamiento por lotes. Ejecuta los comandos que contenga en una terminal.
- **C:** Lenguaje de programación creado en 1972. En el presente contexto se usa para compararlo con el lenguaje PAWN.
- **C#:** Lenguaje que se ha usado para desarrollar la herramienta.
- **DSL:** Domain-Specific Language, Lenguaje específico de dominio. Con este término se califica al lenguaje desarrollado para crear el diagrama y exportarlo a un XML.
- **Evento o Eventos:** Son los desencadenantes de que una secuencia se ejecute. Los eventos son acciones o estados exteriores que detecta la forma de vida artificial.
- **Filo:** Evento que se lanza cuando la cámara de Pleo detecta que existe un filo en el camino. Que Pleo pueda detectarlo depende de la inclinación del cuello.
- **Forma de vida artificial o Formas de vida artificial:** Sistemas artificiales que recrean sistemas de vida reales.
- **Log:** Archivo que guarda las posibles incidencias. En el contexto actual se crea un log de compilación que recoja las posibles incidencias.
- **Obstáculo:** Evento que se lanza cuando la cámara de Pleo detecta que existe un obstáculo en el camino. Que Pleo pueda detectarlo depende de la inclinación del cuello.
- **Orden o Órdenes:** Son las acciones que llevará a cabo la forma de vida artificial cuando un evento se lance.
- **PAWN:** Lenguaje de programación para la forma de vida artificial Pleo.
- **PDK:** Pleo Development kit, conjunto de herramientas para el compilado de archivos para la forma de vida artificial Pleo.
- **Personalidad:** Conjunto de secuencias.
- **PLEO:** Forma de vida artificial para la que se ha desarrollado la herramienta.
- **Plugin:** Extensión que funciona en conjunción con un programa principal y aumenta su funcionalidad.

- **Posado:** Evento que se lanza cuando la forma de vida detecta que ha sido posado.
- **Posición neutra:** Orden que mueve todos los miembros a su posición natura neutra.
- **RAM:** Referido a memoria RAM de un ordenador, es la memoria disponible para la ejecución de programas.
- **Robótica:** ciencia que estudia los robots.
- **Secuencia:** Conjunto ordenado de órdenes que se ejecutan cuando se detecta el evento que define dicha secuencia.
- **SVN:** Sistema de control de versiones que posibilita llevar un control de los cambios realizados en un fichero.
- **UML:** Lenguaje de modelado de sistemas software utilizado para crear y desarrollar diferentes diagramas que detallen el funcionamiento de la herramienta creada.
- **WPF:** Windows Presentation Foundation, tecnología creada por Microsoft para el desarrollo de.
- **XAML:** Lenguaje utilizado para definir interfaces. Utilizado con WPF y C# para desarrollar el presente proyecto.
- **XML:** Metalenguaje extensible de etiquetas. Se utiliza para exportar la personalidad creada.

13.2 Contenido Entregado en el CD-ROM

13.2.1 Contenidos

13.2.1.1 Estructura general directorios del CD

Directorio	Contenido
./ Directorio raíz del CD	Contiene un fichero leeme.txt explicando toda la estructura.
./PleoProg	Contiene toda la estructura de directorios del proyecto para desarrollo. (Ver la tabla Recomendación de estructura de directorios de desarrollo)
./instalacion	Ficheros utilizados para la instalación del proyecto. Estos son: • Directorio Application Files • PleoProg.application • Setup.exe
./documentacion	Contiene toda la documentación asociada al proyecto. • PleoProg.pdf • PleoProg.doc
./documentacion/img	Directorio que contiene las imágenes utilizadas en la documentación.
./documentacion/uml	Ficheros que genera la herramienta Enterprise Architect.
./presentación	Archivo de PowerPoint utilizado en la presentación del proyecto.

13.2.1.2 Estructura de Directorios de PleoProg

Directorio	Contenido
./ Directorio raíz de PleoProg	Contiene los ficheros de proyecto del IDE utilizado. Contiene las carpetas: • PleoProg • src • xml
./PleoProg	Contiene el proyecto completo incluyendo los ficheros que se crean en la compilación.
./xml	Contiene la documentación generada en formato XML.
./src	Raíz de los ficheros fuente. Contiene los siguientes paquetes: • Interfaz • Estructuras • Compilador • GestorPersonalidad
./ayuda	Fichero de ayuda.
./ayuda/html	Ficheros html utilizados para generar la ayuda
./ayuda/img	Imágenes utilizadas para generar la ayuda.
./ayuda/src	Archivos creados por la herramienta que generó la ayuda para PleoProg
./recursos/img	Ficheros utilizados para los gráficos de la aplicación

13.2.2 Código Ejecutable e Instalación

En la carpeta instalación de la raíz del CD se encuentran los archivos necesarios para instalar y ejecutar la herramienta. Ejecutar Setup.exe para iniciar este proceso.

Para que la herramienta pueda probarse completamente es necesario descargar el PDK de la página de Ugobe tal como se explica en el manual.

13.3 Índice Alfabético

P

R

S

U

W

X

Y

13.4 Código Fuente

13.4.1 PleoProg

13.4.1.1 MainWindow.xaml.cs

```
using System;
using System.IO;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Forms;
using System.Windows.Input;
using PleoProg.Compilador;
using PleoProg.Datos;
using PleoProg.Estructuras;
using PleoProg.GestorPersonalidad;
using PleoProg.Interfaz;
using PleoProg.Interfaz.Controles;
using System.Diagnostics;


namespace PleoProg
{
    /// <summary>
    /// Ventana principal de la aplicación
    /// </summary>
    public partial class MainWindow : Window
    {
        /// <summary>
        /// Variable GestorDSL para controlar el dibujado y Personalidad
        /// </summary>
        private GestorDSL gestor;

        /// <summary>
        /// Nombre del proyecto actual
        /// </summary>
        private string sNombreProyecto;

        /// <summary>
        /// Ruta del proyecto actual
        /// </summary>
        private string sRutaProyecto;

        /// <summary>
        /// Constructor de la clase. Inicializa los componentes.
        /// </summary>
        public MainWindow()
        {
            InitializeComponent();
            sRutaProyecto = "";
            inicio();
        }


        /// <summary>
        /// Al hacer clic sobre un botón se resetea el estado de los demás
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonEvento_Click(object sender, RoutedEventArgs e)
        {
            resetBotones(buttonEvento);
        }

        /// <summary>
        /// Al hacer clic sobre un botón se resetea el estado de los demás
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonPosNormal_Click(object sender, RoutedEventArgs e)
```

```
        {
            resetBotones(buttonPosNormal);
        }

        /// <summary>
        /// Al hacer clic sobre un botón se resetea el estado de los demás
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonMov_Click(object sender, RoutedEventArgs e)
        {
            resetBotones(buttonMov);
        }

        /// <summary>
        /// Al hacer clic sobre un botón se resetea el estado de los demás
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCam_Click(object sender, RoutedEventArgs e)
        {
            resetBotones(buttonCam);
        }

        /// <summary>
        /// Al hacer clic sobre un botón se resetea el estado de los demás
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonRec_Click(object sender, RoutedEventArgs e)
        {
            resetBotones(buttonRec);
        }

        /// <summary>
        /// Al hacer clic sobre un botón se resetea el estado de los demás
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonRep_Click(object sender, RoutedEventArgs e)
        {
            resetBotones(buttonRep);
        }

        /// <summary>
        /// Al hacer clic sobre un botón se resetea el estado de los demás
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonEsp_Click(object sender, RoutedEventArgs e)
        {
            resetBotones(buttonEsp);
        }

        /// <summary>
        /// Al hacer clic sobre un botón se resetea el estado de los demás
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonMotion_Click(object sender, RoutedEventArgs e)
        {
            resetBotones(buttonMotion);
        }

        /// <summary>
        /// Al hacer clic sobre el botón compilar se verifica que sea posible y se
procede a llamar a la parte del compilador
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCompile_Click(object sender, RoutedEventArgs e)
        {
            if (gestor.personalidad.iSize() == 0)
            {
                System.Windows.MessageBox.Show(this, "Debe crear un diagrama para
compilarlo", "Compilar", MessageBoxButton.OK, MessageBoxImage.Exclamation);
                return;
            }
```

```
            //Se obtiene una carpeta temporal donde albergar todos los archivos
temporales
            string sRutaTemp = Path.GetTempPath();
            DateTime dtCurrent = DateTime.Now;
            sRutaTemp += "PLEO_" + dtCurrent.Hour + dtCurrent.Minute + dtCurrent.Second
+ "_" + dtCurrent.Day + dtCurrent.Month + dtCurrent.Year + "\\";
            string sRutaXML = sRutaTemp + sNombreProyecto + ".xml";
            Directory.CreateDirectory(sRutaTemp);

            FileStream fS = new FileStream(sRutaXML, FileMode.Create, FileAccess.Write);
            StreamWriter sW = new StreamWriter(fS);
            gestor.personalidad.toXML(sW);
            sW.Flush();
            sW.Close();

            WindowCompilador WCompilador = new WindowCompilador(sRutaXML,
sNombreProyecto, sRutaTemp);
            WCompilador.ShowDialog();
        }



        /// <summary>
        /// Establece el estado de los botones
        /// </summary>
        /// <param name="t"></param>
        private void resetBotones(System.Windows.Controls.Primitives.ToggleButton t)
        {
            //Si el botón que se acaba de tocar es para desactivar no se hace nada
            if (t.IsChecked == false)
                return;
            //Si se está activando algún botón, se desactivan todos
            buttonEvento.IsChecked = false;
            buttonPosNormal.IsChecked = false;
            buttonMov.IsChecked = false;
            buttonCam.IsChecked = false;
            buttonRec.IsChecked = false;
            buttonRep.IsChecked = false;
            buttonEsp.IsChecked = false;
            buttonMotion.IsChecked = false;
            //Se activa el botón deseado
            t.IsChecked = true;
        }

        /// <summary>
        /// Según el botón que esté activada devuelve la orden a crear
        /// </summary>
        /// <returns></returns>
        private int botonOrden()
        {
            if (buttonPosNormal.IsChecked == true)
                return 1;
            if (buttonMov.IsChecked == true)
                return 2;
            if (buttonCam.IsChecked == true)
                return 3;
            if (buttonRec.IsChecked == true)
                return 4;
            if (buttonRep.IsChecked == true)
                return 5;
            if (buttonEsp.IsChecked == true)
                return 6;
            if (buttonEvento.IsChecked == true)
                return 7;
            if (buttonMotion.IsChecked == true)
                return 8;
            return 0;
        }

        /// <summary>
        /// Al pulsar sobre el canvas y levantar la pulsación
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void canvasPrncpl_MouseLeftButtonUp(object sender, MouseButtonEventArgs
e)
        {
```

```
            int iBotonPulsado = botonOrden();
            if (iBotonPulsado == 0)
                return;

            if (gestor.ordenSeleccionada != null)
            {
                gestor.ordenSeleccionada.deseleccionar();
                gestor.ordenSeleccionada = null;
            }
            if (gestor.eventoSeleccionado != null)
            {
                gestor.eventoSeleccionado.deseleccionar();
                gestor.eventoSeleccionado = null;
            }

            Point ptPosCanvas = e.GetPosition(canvasPrncpl);
            Secuencia secActual;

            //Si se desea añadir un evento, se muestra una ventana para verificarlo
            if (iBotonPulsado == 7)
            {
                Evento ev = new Evento();
                WindowEvento WE = new WindowEvento(ev, gestor);
                //Si se cancela no se guarda nada
                if (WE.ShowDialog() == false)
                    return;


                //Si no está repetido guardar y dibujar
                else
                {
                    secActual = gestor.personalidad.insertarEvento(ev);
                    gestor.dibujarControl(ptPosCanvas, secActual, ev);
                }
                return;
            }

            //Si se desea añadir una orden...
            //Si no hay ningún evento se necesita uno
            if (iBotonPulsado != 7 && iBotonPulsado != 0 && gestor.personalidad.iSize()
== 0)
            {
                System.Windows.MessageBox.Show(this, "No es posible añadir una orden si
no existe un evento que la lance primero", "Insertar una orden", MessageBoxButton.OK,
MessageBoxImage.Exclamation);
                return;
            }

            if (gestor.eventoActual == null)
            {
                System.Windows.MessageBox.Show(this, "Seleccione un evento primero",
"Insertar una orden", MessageBoxButton.OK, MessageBoxImage.Exclamation);
                return;
            }

            secActual = gestor.personalidad.buscarSecuencia(gestor.eventoActual.evento);

            int iId = gestor.personalidad.iAcumulador;
            int iCont = secActual.getOrdenes().Count + 1;

            Orden orden = null;

            //Se muestra la ventana correspondiente al tipo de orden que se desea añadir
            switch (iBotonPulsado)
            {
                case 1:
                    orden = new Neutro(iId, iCont);
                    break;
                case 2:
                    orden = new Movimiento(iId, iCont, 0, 0, 1, 0, false);
                    WindowMovimiento WM = new WindowMovimiento((Movimiento)orden);
                    if (WM.ShowDialog() == false)
                        orden = null;
                    break;
                case 3:
                    orden = new CapturaImagen(iId, iCont, "");
                    WindowNombreArchivoOrden WNA = new WindowNombreArchivoOrden(orden);
                    if (WNA.ShowDialog() == false)
```

```
                        orden = null;
                    break;
                case 4:
                    orden = new CapturaSonido(iId, iCont, "", 0);
                    WindowCapturaSonido WCS = new WindowCapturaSonido(orden);
                    if (WCS.ShowDialog() == false)
                        orden = null;
                    break;
                case 5:
                    OpenFileDialog oFD = new OpenFileDialog();

                    string sRutaMedia = Properties.Settings.Default.sCompiladorPleo +
"\\media\\sounds\\";
                    if (Properties.Settings.Default.sCompiladorPleo == "")
                        sRutaMedia =
System.IO.Path.GetPathRoot(System.IO.Path.GetTempPath());

                    oFD.InitialDirectory = "" + sRutaMedia;
                    oFD.Filter = "Waveform Audio (*.wav) | *.wav";
                    oFD.FilterIndex = 0;
                    oFD.RestoreDirectory = true;
                    oFD.ShowDialog();

                    if (oFD.FileName != "")
                    {
                        string sRutaArchivoSonido = oFD.FileName;

                        FileInfo fInfo = new FileInfo(sRutaArchivoSonido);
                        long lSize = fInfo.Length;

                        if (lSize > 1048576)
                        {
                            System.Windows.MessageBox.Show(this, "Para preservar la
memoria no se permiten archivos de audio de más de 1 MB de tamaño", "Insertar
reproducción de sonido", MessageBoxButton.OK, MessageBoxImage.Exclamation);
                            orden = null;
                        }
                        else
                            orden = new ReproducirSonido(iId, iCont,
sRutaArchivoSonido);
                    }
                    else
                        orden=null;

                    break;
                case 6:
                    orden = new Espera(iId, iCont, 0);
                    WindowTiempoEspera WEspera = new WindowTiempoEspera((Espera)orden);
                    if (WEspera.ShowDialog() == false)
                        orden = null;
                    break;
                case 8:
                    OpenFileDialog oFD2 = new OpenFileDialog();


                    string sRutaMotion = Properties.Settings.Default.sCompiladorPleo +
"\\media\\motions\\";
                    if (Properties.Settings.Default.sCompiladorPleo == "")
                        sRutaMotion =
System.IO.Path.GetPathRoot(System.IO.Path.GetTempPath());

                    oFD2.InitialDirectory = "" + sRutaMotion;
                    oFD2.Filter = "Motion File (*.csv) | *.csv";
                    oFD2.FilterIndex = 0;
                    oFD2.RestoreDirectory = true;
                    oFD2.ShowDialog();

                    if (oFD2.FileName != "")
                    {
                        string sRutaArchivoMotion = oFD2.FileName;
                        FileInfo fInfo = new FileInfo(sRutaArchivoMotion);
                        long lSize = fInfo.Length;

                        if (lSize > 1048576)
                        {
                            System.Windows.MessageBox.Show(this, "Para preservar la
memoria no se permiten archivos de animaciones de más de 1 MB de tamaño", "Insertar
reproducción de animación", MessageBoxButton.OK, MessageBoxImage.Exclamation);
```

```
                            orden = null;
                        }
                        else
                            orden = new Motion(iId, iCont, sRutaArchivoMotion);
                    }
                    else
                        orden = null;
                    break;

            }
            //Si no se ha modificado la orden se vuelve sin guardar cambios
            if (orden == null)
                return;

            //Se inserta la orden en la secuencia
            secActual.insertaOrden(orden);
            //Se dibuja el control
            gestor.dibujarControl(ptPosCanvas, secActual, orden);
            //Se incrementa el acumulador de ordenes
            gestor.personalidad.iAcumulador++;
        }

        /// <summary>
        /// Cuando una orden o evento está seleccionado y se pulsa sobre eliminar se
elimina
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonEliminar_Click(object sender, RoutedEventArgs e)
        {
            ControlEvento cE = gestor.eventoSeleccionado;
            ControlOrden cO = gestor.ordenSeleccionada;

            if (cO != null)
            {
                Orden orden = cO.orden;
                Secuencia sec = cO.secuencia;
                gestor.eliminarControlInterfaz(cO);
                sec.eliminarOrden(orden);
                return;
            }

            if (cE != null)
            {
                Secuencia sec = cE.secuencia;
                if (sec.getOrdenes().Count != 0)
                {
                    //Se muestra un error y se advierte que se eliminaran las ordenes
asociadas de continuar
                    if ((System.Windows.MessageBox.Show(this, "El evento tiene ordenes
asociadas, si continua borrará el evento y todas las ordenes asociadas ¿desea
continuar?", "Eliminar un evento", MessageBoxButton.YesNo, MessageBoxImage.Question)) ==
MessageBoxResult.No)
                        return;
                }
                gestor.eliminarControlInterfaz(cE);
                gestor.personalidad.eliminarEvento(cE.evento);
                return;
            }
        }

        /// <summary>
        /// Al pulsar una tecla en la aplicación...
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void gridPrincpl_KeyDown(object sender,
System.Windows.Input.KeyEventArgs e)
        {
            //Si noes la tecla borrar se elimina la orden o evento, si procede
            if ((e.Key == Key.Delete || e.Key == Key.Back))
                buttonEliminar_Click(sender, e);


        }

        /// <summary>
        /// Se ejecuta al pulsar sobre el botón de incrementar turno
```

```
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonSubirTurno_Click(object sender, RoutedEventArgs e)
        {
            ControlOrden cO = gestor.ordenSeleccionada;

            if (cO != null)
                gestor.incrementarTurno(cO);
        }

        /// <summary>
        /// Se ejecuta al pulsar sobre el botón de derecmentar turno
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonDecrementarTurno_Click(object sender, RoutedEventArgs e)
        {
            ControlOrden cO = gestor.ordenSeleccionada;

            if (cO != null)
                gestor.decrementarTurno(cO);
        }


        /// <summary>
        /// Al seleccionar un evento en el combobox de evento actual...
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void comboBoxEventoActual_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            if (gestor.ordenSeleccionada != null)
            {
                gestor.ordenSeleccionada.deseleccionar();
                gestor.ordenSeleccionada = null;
            }
            if (gestor.eventoSeleccionado != null)
            {
                gestor.eventoSeleccionado.deseleccionar();
                gestor.eventoSeleccionado = null;
            }


            string sEv = (string)comboBoxEventoActual.SelectedItem;
            if (sEv == null)
                return;

            int iEv = Util.getCodigoEvento(sEv);
            gestor.eventoActual = gestor.buscarControlEvento(new Evento(iEv));
            gestor.eventoSeleccionado = gestor.eventoActual;
            gestor.eventoSeleccionado.seleccionar();

        }

        /// <summary>
        /// Crea una nueva personalidad
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void nuevo_Click(object sender, RoutedEventArgs e)
        {
            string sNombre = null;
            //Nuevo cuadro para introducir nombre
            WindowNombreArchivo WNA = new WindowNombreArchivo(sNombre);

            //Si es correcto activar todo
            if (WNA.ShowDialog() == false)
                return;
            sNombre = WNA.sNombre;
            if (sNombre == null || sNombre == "")
                return;

            reiniciarCanvas();
            gestor = new GestorDSL(canvasPrncpl, comboBoxEventoActual);
            this.sNombreProyecto = sNombre;
        }
```

```
        /// <summary>
        /// Sale de la aplicación
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void salir_Click(object sender, RoutedEventArgs e)
        {
            //Se pregunta si se desea guardar antes de cargar un nuevo proyecto
            if ((System.Windows.MessageBox.Show(this, "¿Desea guardar el proyecto
actual?", "Salir", MessageBoxButton.YesNo, MessageBoxImage.Question)) ==
MessageBoxResult.Yes)
                guardar_Click(sender, e);

            this.Close();
        }

        /// <summary>
        /// Cierra la edición de la personalidad actual
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void cerrar_Click(object sender, RoutedEventArgs e)
        {
            //Se pregunta si se desea guardar antes de cargar un nuevo proyecto
            if ((System.Windows.MessageBox.Show(this, "¿Desea guardar el proyecto
actual?", "Cerrar Actual Proyecto", MessageBoxButton.YesNo, MessageBoxImage.Question))
== MessageBoxResult.Yes)
                guardar_Click(sender, e);

            reiniciarCanvas();
            inicio();
        }

        /// <summary>
        /// Guarda la perosnalidad actual
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void guardar_Click(object sender, RoutedEventArgs e)
        {
            Microsoft.Win32.SaveFileDialog sFD = new Microsoft.Win32.SaveFileDialog();
            sFD.FileName = sNombreProyecto + ".bin";
            sFD.DefaultExt = ".bin";

            if (sRutaProyecto == "")
                sFD.InitialDirectory =
System.IO.Path.GetPathRoot(System.IO.Path.GetTempPath());
            else
                sFD.InitialDirectory = System.IO.Path.GetDirectoryName(sRutaProyecto);

            sFD.Title = "Guardar como...";
            sFD.CheckPathExists = true;
            sFD.Filter = "BIN File (*.bin) | *.bin";

            if (sFD.ShowDialog() == false)
                return;

            sRutaProyecto = sFD.FileName;
            guardarProyecto(sRutaProyecto);
            sNombreProyecto = System.IO.Path.GetFileNameWithoutExtension(sRutaProyecto);
            this.Title = "PleoProg-" + sNombreProyecto;
        }

        /// <summary>
        /// Carga una personalidad previamente creada
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void cargar_Click(object sender, RoutedEventArgs e)
        {
            GestorDSL gestorAntiguo = gestor;

            //Se pregunta si se desea guardar antes de cargar un nuevo proyecto
            if ((System.Windows.MessageBox.Show(this, "¿Desea guardar el proyecto
actual?", "Cargar Nuevo Proyecto", MessageBoxButton.YesNo, MessageBoxImage.Question)) ==
MessageBoxResult.Yes)
                guardar_Click(sender, e);
```

```
            OpenFileDialog oFD = new OpenFileDialog();

            if (sRutaProyecto == "")
                oFD.InitialDirectory =
System.IO.Path.GetPathRoot(System.IO.Path.GetTempPath());
            else
                oFD.InitialDirectory = sRutaProyecto;

            oFD.Filter = "BIN File (*.bin) | *.bin";
            oFD.FilterIndex = 0;
            oFD.RestoreDirectory = true;
            oFD.ShowDialog();

            if (oFD.FileName != "")
            {
                sRutaProyecto = oFD.FileName;
                if (!cargarProyecto(sRutaProyecto))
                {
                    gestor = gestorAntiguo;
                    gestor.dibujarPersonalidad();
                }
            }
            else
                return;

            this.Title = "PleoProg-" + sNombreProyecto;
        }

        /// <summary>
        /// Muestra una ventana con las opciones de inicio
        /// </summary>
        private void inicio()
        {
            WindowOpcionesArchivo WOA = new WindowOpcionesArchivo();
            if (WOA.ShowDialog() == true)
            {
                gestor = new GestorDSL(canvasPrncpl, comboBoxEventoActual);

                string sOpcion = WOA.sOpcion;
                if (WOA.RBNuevo.IsChecked == true)
                    sNombreProyecto = sOpcion;
                else
                    cargarProyecto(sOpcion);
            }
            else
                this.Close();

            this.Title = "PleoProg-" + sNombreProyecto;
        }

        /// <summary>
        /// Carga un proyecto
        /// </summary>
        /// <param name="sRuta">Ruta del proyecto</param>
        /// <returns>true si es correcto</returns>
        private bool cargarProyecto(string sRuta)
        {
            try
            {
                sRutaProyecto = sRuta;
                reiniciarCanvas();
                gestor = new GestorDSL(canvasPrncpl, comboBoxEventoActual);
                gestor.abrirPersonalidad(sRutaProyecto);
                sNombreProyecto = System.IO.Path.GetFileNameWithoutExtension(sRuta);
            }
            catch (Exception)
            {
                System.Windows.MessageBox.Show(this, "Ha ocurrido un error inesperado.
Es posible que el archivo " + sRuta + " que intenta cargar se encuentre dañado", "Cargar
Proyecto", MessageBoxButton.OK, MessageBoxImage.Error);
                return false;
            }
            return true;
        }

        /// <summary>
        /// Guarda un proyecto
```

```
        /// </summary>
        /// <param name="sRuta">ruta del proyecto</param>
        /// <returns>true si es correcto</returns>
        private bool guardarProyecto(string sRuta)
        {
            try
            {
                gestor.guardarPersonalidad(sRuta);
            }
            catch (Exception)
            {
                System.Windows.MessageBox.Show(this, "Ha ocurrido un error inesperado.
Es posible que la ubicación: " + sRuta + " se encuentre en uso o no sea correcta",
"Guardar Proyecto", MessageBoxButton.OK, MessageBoxImage.Error);
                return false;
            }
            return true;
        }

        /// <summary>
        /// Se reinician todos los elementos visuales
        /// </summary>
        private void reiniciarCanvas()
        {
            comboBoxEventoActual.Items.Clear();
            canvasPrncpl.Children.Clear();
        }

        private void menuAcerca_Click(object sender, RoutedEventArgs e)
        {
            WindowAcerca acerca = new WindowAcerca(this);
            acerca.ShowDialog();
        }

        private void menuPagina_Click(object sender, RoutedEventArgs e)
        {
            Process.Start("http://www.davidgg.es/PleoProg");
        }

        /// <summary>
        /// Muestra una ventana con las preferencias del programa
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void menuPreferencias_Click(object sender, RoutedEventArgs e)
        {
            WindowPreferencias wP = new WindowPreferencias();
            wP.ShowDialog();
        }

        private void menuAyuda_Click(object sender, RoutedEventArgs e)
        {
            System.Windows.Forms.Help.ShowHelp(null,
System.Windows.Forms.Application.StartupPath + @"\Recursos\PleoProgHelp.chm");
        }


    }
}
```

13.4.1.2 MainWindow.xaml

```
<Window
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="d"
x:Class="PleoProg.MainWindow"
        xmlns:my="clr-namespace:PleoProg.Controles"
        Title="PleoProg" Height="600" Width="800" MinWidth="800" MinHeight="600"
WindowStartupLocation="CenterOwner" WindowStyle="SingleBorderWindow"
Icon="/PleoProg;component/Recursos/dinosauro.png">

    <Window.CommandBindings>
        <CommandBinding Command='ApplicationCommands.Help' Executed='menuAyuda_Click' />
    </Window.CommandBindings>

    <Grid Background="Gainsboro">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="753*" />
            <ColumnDefinition Width="25*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="22" />
            <RowDefinition Height="60" />
            <RowDefinition Height="30" />
            <RowDefinition Height="473*" />
            <RowDefinition Height="26" />
        </Grid.RowDefinitions>

        <Menu  IsMainMenu="True" HorizontalAlignment="Stretch" VerticalAlignment="Top"
MinHeight="23" x:Name="menu1" Grid.ColumnSpan="2" Grid.RowSpan="2">
            <MenuItem Header="Archivo">
                <MenuItem Header="Nuevo..." Name="menuNuevo" Click="nuevo_Click" />
                <MenuItem Header="Cargar..." Name="menuCargar" Click="cargar_Click" />
                <Separator />
                <MenuItem Header="Guardar..." Name="menuGuardar" Click="guardar_Click"
/>
                <MenuItem Header="Cerrar Actual" Name="menuCerrar" Click="cerrar_Click"
/>
                <Separator />
                <MenuItem Header="Salir" Name="menuSalir" Click="salir_Click" />
            </MenuItem>

            <MenuItem Header="Edición" >
                <MenuItem Header="Aumentar Turno" Name="menuAumentarTurno"
Click="buttonSubirTurno_Click" />
                <MenuItem Header="Disminuir Turno" Name="menuDisminuirTurno"
Click="buttonDecrementarTurno_Click" />
                <MenuItem Header="Eliminar" Name="menuEliminar"
Click="buttonEliminar_Click" />
            </MenuItem>

            <MenuItem Header="Preferencias" Click="menuPreferencias_Click" />

            <MenuItem Header="Ayuda">
                <MenuItem Header="Ver Ayuda" Name="menuAyuda" Click="menuAyuda_Click"/>
                <Separator />
                <MenuItem Header="Página de PleoProg" Name="menuPagina"
Click="menuPagina_Click" />
                <MenuItem Header="Acerca de PleoProg" Name="menuAcerca"
Click="menuAcerca_Click" />
            </MenuItem>

        </Menu>
        <ToolBarTray HorizontalAlignment="Stretch" Grid.Row="1"  VerticalAlignment="Top"
MinHeight="60" Grid.ColumnSpan="2" Height="Auto">
            <ToolBar x:Name="menuEvento" Width="Auto" HorizontalAlignment="Stretch"
MinHeight="40" Grid.Column="1">
                <ToggleButton x:Name="buttonEvento" HorizontalAlignment="Left"
VerticalAlignment="Center" Click="buttonEvento_Click">
                    <Image Name="ImgEvento" Source="/Recursos/IcoEvento.png" Height="30"
Width="30" ToolTip="Insertar Evento" />
                </ToggleButton>
            </ToolBar>
```

```
            <ToolBar  x:Name="menuTools" Width="Auto" HorizontalAlignment="Stretch"
MinHeight="60" Grid.Column="0">

                <ToggleButton x:Name="buttonPosNormal" Margin="5,0,0,0"
HorizontalAlignment="Left" VerticalAlignment="Center" Click="buttonPosNormal_Click"
ToolTip="Posición normal">
                    <Image Source="/Recursos/IcoNeutro.png" Height="30" Width="30" />
                </ToggleButton>

                <ToggleButton x:Name="buttonMov" Margin="5,0,0,0"
HorizontalAlignment="Left" VerticalAlignment="Center" Click="buttonMov_Click"
ToolTip="Orden de Movimiento">
                    <Image Source="/Recursos/IcoMov.png" Height="30" Width="30"  />
                </ToggleButton>

                <ToggleButton x:Name="buttonCam" Margin="5,0,0,0"
HorizontalAlignment="Left" VerticalAlignment="Center" Click="buttonCam_Click"
ToolTip="Orden de Captura de Imagen">
                    <Image Source="/Recursos/IcoFoto.png" Height="30" Width="30"  />
                </ToggleButton>

                <ToggleButton x:Name="buttonRec" Margin="5,0,0,0"
HorizontalAlignment="Left" VerticalAlignment="Center" Click="buttonRec_Click"
ToolTip="Orden de Captura Sonido">
                    <Image Source="/Recursos/IcoRec.png" Height="30" Width="30"  />
                </ToggleButton>

                <ToggleButton x:Name="buttonRep" Margin="5,0,0,0"
HorizontalAlignment="Left" VerticalAlignment="Center" Click="buttonRep_Click"
ToolTip="Orden de Reproducir Sonido">
                    <Image Source="/Recursos/IcoSonido.png" Height="30" Width="30" />
                </ToggleButton>

                <ToggleButton x:Name="buttonEsp" Margin="5,0,0,0"
HorizontalAlignment="Left" VerticalAlignment="Center" Click="buttonEsp_Click"
ToolTip="Orden de Tiempo en Espera">
                    <Image Source="/Recursos/IcoTiempo.png" Height="30" Width="30"  />
                </ToggleButton>

            </ToolBar>

            <ToolBar x:Name="menuMovEspecial" Width="Auto" HorizontalAlignment="Stretch"
MinHeight="40" Grid.Column="1">
                <ToggleButton x:Name="buttonMotion" HorizontalAlignment="Left"
VerticalAlignment="Center" Click="buttonMotion_Click">
                    <Image Source="/Recursos/IcoMotion.png" Height="30" Width="30"
ToolTip="Animaciones" />
                </ToggleButton>
            </ToolBar>

            <ToolBar x:Name="menuEditar" Width="Auto" HorizontalAlignment="Stretch"
MinHeight="40" Grid.Column="1">
                <Button x:Name="buttonIncrementarTurno" Margin="5,0,0,0"
HorizontalAlignment="Left" VerticalAlignment="Center" Click="buttonSubirTurno_Click">
                    <Image Source="/Recursos/IcoIncrementar.png" Height="30" Width="30"
ToolTip="Incrementar Turno" />
                </Button>
                <Button x:Name="buttonDecrementarTurno" Margin="5,0,0,0"
HorizontalAlignment="Left" VerticalAlignment="Center"
Click="buttonDecrementarTurno_Click">
                    <Image Source="/Recursos/IcoDecrementar.png" Height="30" Width="30"
ToolTip="Decrementar Turno" />
                </Button>
                <Button x:Name="buttonEliminarOrden" Margin="5,0,0,0"
HorizontalAlignment="Left" VerticalAlignment="Center" Click="buttonEliminar_Click">
                    <Image Source="/Recursos/IcoEliminar.png" Height="30" Width="30"
ToolTip="Eliminar Orden" />
                </Button>
            </ToolBar>

            <ToolBar x:Name="menuExport" Width="Auto" HorizontalAlignment="Stretch"
MinHeight="40" Grid.Column="1">
                <Button x:Name="buttonCompile" HorizontalAlignment="Left"
VerticalAlignment="Center" Click="buttonCompile_Click">
                    <Image Source="/Recursos/IcoToPleo.png" Height="30" Width="30"
ToolTip="Compilar Programa" />
                </Button>
            </ToolBar>
```

```
        </ToolBarTray>

        <Grid Name="gridPrincpl" Grid.Row="3"  Grid.ColumnSpan="2" Grid.RowSpan="2"
KeyDown="gridPrincpl_KeyDown">
            <ScrollViewer HorizontalScrollBarVisibility="Auto"
VerticalScrollBarVisibility="Auto">
                <my:ScrollableCanvas HorizontalAlignment="Stretch" x:Name="canvasPrncpl"
VerticalAlignment="Stretch" Grid.Row="2"
MouseLeftButtonUp="canvasPrncpl_MouseLeftButtonUp" AllowDrop="True" Grid.ColumnSpan="2"
Grid.RowSpan="1" Focusable="True">
                    <my:ScrollableCanvas.Background>
                        <SolidColorBrush />
                    </my:ScrollableCanvas.Background>
                </my:ScrollableCanvas>
            </ScrollViewer>
        </Grid>

        <Grid Background="WhiteSmoke"  Grid.Row="2"  Grid.ColumnSpan="2"
Grid.RowSpan="1" HorizontalAlignment="Stretch" Width="Auto" MinHeight="30">
            <Label Grid.Row="2" Grid.Column="0" Height="25" Width="100"
HorizontalAlignment="Left" Margin="5,0,0,0" Content="Evento Actual"
VerticalAlignment="Center" />
            <ComboBox Grid.Row="2" Grid.Column="0" Height="25" Width="300"
HorizontalAlignment="Left" Margin="115,0,0,0" Name="comboBoxEventoActual"
VerticalAlignment="Center" FontSize="12"
SelectionChanged="comboBoxEventoActual_SelectionChanged" ItemsSource="{Binding}" />
        </Grid>
    </Grid>
</Window>
```

13.4.2 PleoProg.GestorPersonalidad

13.4.2.1 Personalidad.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PleoProg.Estructuras;
using System.IO;

namespace PleoProg.GestorPersonalidad
{
    /// <summary>
    /// Clase que almacena los valores de una personalidad
    /// </summary>
    [Serializable]
    public class Personalidad
    {
        /// <summary>
        /// Constructor de la clase Personalidad.
        /// </summary>
        public Personalidad()
        {
            lsSecuencias = new List<Secuencia>();
            iAcumulador = 0;
        }

        /// <summary>
        /// Almacena las diferences secuencias de la personalidad
        /// </summary>
        private List<Secuencia> lsSecuencias;

        /// <summary>
        /// Proporciona un acumulador que sirve de identificador unico para todas las
ordenes
        /// </summary>
        public int iAcumulador
        {
            get;
```

```
            set;
        }

        /// <summary>
        /// Devuelve la lista de secuencias
        /// </summary>
        /// <returns>Lista de secuencias</returns>
        public List<Secuencia> getSecuencias()
        {
            return lsSecuencias;
        }

        /// <summary>
        /// Devuelve el número de secuencias guardadas hasta el momento.
        /// Este es el mismo número de eventos.
        /// </summary>
        /// <returns>Número de secuencias</returns>
        public int iSize()
        {
            return lsSecuencias.Count;
        }


        /// <summary>
        /// Inserta un evento en la secuencia
        /// </summary>
        /// <param name="ev">Evento a insertar</param>
        /// <returns>Devuelve true si la operación se completa con éxito, devuelve false
si ya existía un evento igual</returns>
        public Secuencia insertarEvento(Evento ev)
        {
            foreach (Secuencia s in lsSecuencias)
                if (s.evento.Equals(ev))
                    return null;

            Secuencia sec = new Secuencia(ev);
            lsSecuencias.Add(sec);
            return sec;
        }

        /// <summary>
        /// Modifica un evento
        /// </summary>
        /// <param name="oldEv">Antiguo evento</param>
        /// <param name="newEv">Nuevo evento</param>
        /// <returns>true si se ha modificado correctamente</returns>
        public bool editarEvento(Evento oldEv, Evento newEv)
        {
            foreach (Secuencia s in lsSecuencias)
                if (s.evento.Equals(oldEv))
                {
                    s.editarEvento(newEv);
                    return true;
                }

            return false;
        }

        /// <summary>
        /// Elimina un evento y todas sus ordenes
        /// </summary>
        /// <param name="ev">Evento a eliminar</param>
        /// <returns>true si se ha eliminado con éxito</returns>
        public bool eliminarEvento(Evento ev)
        {
            foreach (Secuencia s in lsSecuencias)
                if (s.evento.Equals(ev))
                {
                    lsSecuencias.Remove(s);
                    return true;
                }
            return false;
        }


        /// <summary>
        /// Busca una determinada secuencia atendiendo al evento que la inicia
        /// </summary>
```

```
        /// <param name="ev">Evento de la secuencia</param>
        /// <returns>Secuencia del evento</returns>
        public Secuencia buscarSecuencia(Evento ev)
        {
            foreach (Secuencia s in lsSecuencias)
                if (s.evento.Equals(ev))
                    return s;
            return null;
        }

        /// <summary>
        /// Busca una determinada secuencia atendiendo a una orden que se ejecuta en
dicha secuencia
        /// </summary>
        /// <param name="orden">Orden de la secuencia</param>
        /// <returns>Secuencia donde está la orden</returns>
        public Secuencia buscarSecuencia(Orden orden)
        {
            foreach (Secuencia s in lsSecuencias)
                if (s.existe(orden))
                    return s;
            return null;
        }

        /// <summary>
        /// Escribe en el flujo que se le pase un trozo de código XML correspondiente a
la personalidad
        /// </summary>
        /// <param name="sW">StreamWriter donde se escribirá la información</param>
        public void toXML(StreamWriter sW)
        {
            sW.WriteLine("<?xml version=\"1.0\" encoding=\"UTF-8\"?> ");
            sW.WriteLine("<Personalidad>");
            foreach (Secuencia s in lsSecuencias)
            {
                s.toXML(sW);
            }
            sW.WriteLine("</Personalidad>");
        }
    }
}
```

13.4.2.2 Secuencia.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using PleoProg.Estructuras;
using System.IO;


namespace PleoProg.GestorPersonalidad
{
    /// <summary>
    /// Clase que contiene una determinada secuencia con sus ordenes
    /// </summary>
    [Serializable]
    public class Secuencia
    {
        /// <summary>
        /// Constructor de la clase Secuencia
        /// </summary>
        /// <param name="ev">Evento que dispara la secuencia de órdenes</param>
        public Secuencia(Evento ev)
        {
            this.evento = ev;
        }

        /// <summary>
        /// Lista enlazada con las órdenes que contiene la secuencia
        /// </summary>
        private LinkedList<Orden> lnkLsOrdenes = new LinkedList<Orden>();

        /// <summary>
```

```
        /// Devuelve las órdenes de la secuencia
        /// </summary>
        /// <returns>Lista de ordenes</returns>
        public LinkedList<Orden> getOrdenes()
        {
            return lnkLsOrdenes;
        }

        /// <summary>
        /// Evento que lanza las órdenes de la secuencia
        /// </summary>
        public Evento evento
        {
            get;
            set;
        }

        /// <summary>
        /// Inserta una orden en la secuencia
        /// </summary>
        /// <param name="orden">Orden a insertar</param>
        /// <returns>true si se ha insertado correctamente</returns>
        public bool insertaOrden(Orden orden)
        {
            if (existe(orden) == true)
                return false;
            lnkLsOrdenes.AddLast(orden);
            return true;
        }

        /// <summary>
        /// Elimina una orden de la secuencia
        /// </summary>
        /// <param name="orden">Orden a eliminar</param>
        /// <returns>true si se eliminado correctamente</returns>
        public bool eliminarOrden(Orden orden)
        {
            if (existe(orden) == false)
                return false;

            lnkLsOrdenes.Remove(orden);
            return true;
        }

        /// <summary>
        /// Determina si existe la orden en la secuencia
        /// </summary>
        /// <param name="orden">Orden a buscar</param>
        /// <returns>true si existe</returns>
        public bool existe(Orden orden)
        {
            if (lnkLsOrdenes.Find(orden) == null)
                return false;
            return true;
        }

        /// <summary>
        /// Edita el evento
        /// </summary>
        /// <param name="ev">Nuevo valor de evento</param>
        public void editarEvento(Evento ev)
        {
            this.evento = ev;
        }

        /// <summary>
        /// Devuelve la orden que tenga el turno que se le pase
        /// </summary>
        /// <param name="iTurno">turno de la orden a buscar</param>
        /// <returns>Orden buscada</returns>
        public Orden buscarOrden(int iTurno)
        {
            foreach (Orden o in lnkLsOrdenes)
                if (o.iTurno == iTurno)
                    return o;
            return null;
        }
```

```
        /// <summary>
        /// Escribe en el flujo que se le pase un trozo de código XML correspondiente a
la secuencia
        /// </summary>
        /// <param name="sW">StreamWriter donde se escribirá la información</param>
        public void toXML(StreamWriter sW)
        {
            sW.WriteLine("\t<Secuencia>");
            sW.WriteLine("\t\t<Evento>" + evento.iId + "</Evento>");
            sW.WriteLine("\t\t<Ordenes>");
            foreach (Orden o in lnkLsOrdenes)
            {
                o.toXML(sW);
            }

            sW.WriteLine("\t\t</Ordenes>");
            sW.WriteLine("\t</Secuencia>");
        }
    }
}
```

13.4.3 PleoProg.Compilador

13.4.3.1 Compilador.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace PleoProg.Compilador
{
    /// <summary>
    /// Clase abstracta compilador de la que heredena cada uno de los compiladores de
las formas de vida disponibles
    /// </summary>
    public abstract class Compilador
    {
        /// <summary>
        /// Constructor de la clase Compilador
        /// </summary>
        /// <param name="iFormaVida">Forma de vida</param>
        /// <param name="iVFormaVida">Versión de la forma de vida</param>
        /// <param name="sRutaXML">Ruta del XML de entrada</param>
        /// <param name="sRutaDestino">Ruta del archivo de salida</param>
        /// <param name="sRutaCompilador">Ruta del compilador</param>
        /// <param name="sNombreProyecto">Nombre del proyecto actual</param>
        /// <param name="sRutaTemp">Ruta temporal</param>
        /// <param name="swLog">StreamWriter del log</param>
        public Compilador(int iFormaVida, int iVFormaVida, string sRutaXML, string
sRutaDestino, string sRutaCompilador, string sNombreProyecto, string sRutaTemp,
StreamWriter swLog)
        {
            this.iFormaVida = iFormaVida;
            this.iVFormaVida = iVFormaVida;
            this.sRutaXML = sRutaXML;
            this.sRutaDestino = sRutaDestino;
            this.sRutaCompilador = sRutaCompilador;
            this.sNombreProyecto = sNombreProyecto;
            this.sRutaTemp = sRutaTemp;
            this.swLog = swLog;
        }

        /// <summary>
        /// Entero que indica la forma de vida para la que se está programando
        /// </summary>
        protected int iFormaVida
        {
            get;
            private set;
```

```
        }

        /// <summary>
        /// Entero que indica la versión de la forma de vida para la que se está
programando
        /// </summary>
        protected int iVFormaVida
        {
            get;
            private set;
        }

        /// <summary>
        /// Ruta del archivo XML resultante de la creación del diagrama
        /// </summary>
        protected string sRutaXML
        {
            get;
            private set;
        }

        /// <summary>
        /// Ruta de destino del archivo resultante de la compilación
        /// </summary>
        protected string sRutaDestino
        {
            get;
            private set;
        }

        /// <summary>
        /// Ruta del compilador de la forma de vida
        /// </summary>
        protected string sRutaCompilador
        {
            get;
            set;
        }

        /// <summary>
        /// Nombre del proyecto que se está programando
        /// </summary>
        protected string sNombreProyecto
        {
            get;
            set;
        }

        /// <summary>
        /// Ruta temporal donde se guardan los archivos temporales de la compilación
        /// </summary>
        protected string sRutaTemp
        {
            get;
            set;
        }

        /// <summary>
        /// StreamWriter usado para escribir un log
        /// </summary>
        protected StreamWriter swLog
        {
            get;
            set;
        }

        /// <summary>
        /// Método para el compilado del programa
        /// </summary>
        /// <returns>true si se ha realizado con éxito, false si existe algún
error</returns>
        public abstract bool compilar();

        /// <summary>
        /// Método para verificar que la ruta del compilador de la forma de vida es la
correcta
        /// </summary>
        /// <returns>true si es correcta, false si no lo es</returns>
```

```
        public abstract bool rutaCompiladorValida();
    }
}
```

13.4.3.2 CompiladorPleo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Data;
using System.Xml.Linq;
using System.Xml;
using System.IO;
using PleoProg.Datos;
using System.Xml.XPath;
using System.Diagnostics;

namespace PleoProg.Compilador
{
    /// <summary>
    /// Clase especifica para el compilado en formas de vida artificiales Pleo
    /// </summary>
    class CompiladorPleo : Compilador
    {
        public CompiladorPleo(int iVFormaVida, string sRutaXML, string sRutaDestino,
string sRutaCompilador, string sNombreProyecto, string sRutaTemp, StreamWriter swLog)
            : base(0, iVFormaVida, sRutaXML, sRutaDestino, sRutaCompilador,
sNombreProyecto, sRutaTemp, swLog)
        {
            this.sRutaTemp = sRutaTemp;
        }

        /// <summary>
        /// Indica si se han de adjuntar sonidos
        /// </summary>
        private bool bSonidos
        {
            get;
            set;
        }

        /// <summary>
        /// Indica si se han de adjuntar animaciones
        /// </summary>
        private bool bMotions
        {
            get;
            set;
        }

        /// <summary>
        /// Indica si se ha de crear un método de espera
        /// </summary>
        private bool bEspera
        {
            get;
            set;
        }

        /// <summary>
        /// Indica si el evento alzado o posado se tiene k insertar
        /// </summary>
        private bool bAlzadoPosado
        {
            get;
            set;
        }

        /// <summary>
        /// Crea en la carpeta temporal una carpeta con el esqueleto de la aplicación
        /// </summary>
        private bool crearEsqueleto()
        {
            bool bCorrecto = true;
```

```
            swLog.WriteLine("Creando árbol de directorios...\n");
            try
            {
                // Se crean el arbol de carpetas para guardar los sonidos
                Directory.CreateDirectory(sRutaTemp + "media");
                swLog.WriteLine("Directorio " + sRutaTemp + "media creado");

                Directory.CreateDirectory(sRutaTemp + "media\\sounds");
                swLog.WriteLine("Directorio " + sRutaTemp + "media\\sounds creado");
                Directory.CreateDirectory(sRutaTemp + "media\\motions");
                swLog.WriteLine("Directorio " + sRutaTemp + "media\\motions creado");
                Directory.CreateDirectory(sRutaTemp + "media\\scripts");
                swLog.WriteLine("Directorio " + sRutaTemp + "media\\scripts creado");
            }

            catch (DirectoryNotFoundException)
            {
                swLog.WriteLine("El directorio temporal " + sRutaTemp + " no está
disponible");
                bCorrecto = false;
            }

            catch (UnauthorizedAccessException)
            {
                swLog.WriteLine("No se poseen los permisos necesarios para escribir en
el directorio " + sRutaTemp);
                bCorrecto = false;
            }

            catch (PathTooLongException)
            {
                swLog.WriteLine("Ruta temporal " + sRutaTemp + " demasiado larga");
                bCorrecto = false;
            }

            catch (Exception ex)
            {
                swLog.WriteLine("Error al intentar crear árbol de directorios: " +
ex.Message);
                bCorrecto = false;
            }

            return bCorrecto;
        }

        /// <summary>
        /// Crea el archivo UPF de la aplicación
        /// </summary>
        private void crearArchivoUPF()
        {
            FileStream fS = new FileStream(sRutaTemp + "\\" + sNombreProyecto + ".upf",
FileMode.Create, FileAccess.Write);
            StreamWriter sW = new StreamWriter(fS);

            // Si la versión de Pleo es la 1.0
            if (iVFormaVida == 10)
            {
                sNombreProyecto = "pleo";
                sW.WriteLine("<ugobe_project name=\"" + sNombreProyecto + "\">");
                sW.WriteLine("<options>");
                sW.WriteLine("<set name=\"top\" value=\"../..\" />");
                sW.WriteLine("<include value=\"./include:${top}/include\" />");
                sW.WriteLine("<tools>");
                sW.WriteLine("<pawn value=\"pawncc32 %i -O1 -S64 -v2 -C- %I TARGET=100 -
o%o\" />");
                sW.WriteLine("<block value=\"pleocc -b512 -v %i\" />");
                sW.WriteLine("</tools>");
                sW.WriteLine("<directories>");
                sW.WriteLine("<build value=\"build\" />");
                sW.WriteLine("<include value=\"include\" />");
                sW.WriteLine("</directories>");
                sW.WriteLine("<umf value=\"3\"/>");
                sW.WriteLine("<folders />");
                sW.WriteLine("<block />");
            }

            // Si la versión de Pleo es la 1.1
            if (iVFormaVida == 11)
```

```
            {

                sW.WriteLine("<ugobe_project name=\"" + sNombreProyecto + "\">");
                sW.WriteLine("<options>");
                sW.WriteLine("<set name=\"top\" value=\"../..\" />");
                sW.WriteLine("<include value=\"./include:${top}/include\" />");
                sW.WriteLine("<tools>");
                sW.WriteLine("<pawn value=\"pawncc %i -V2048 -O2 -S64 -v2 -C- %I
TARGET=100 -o%o\" />");
                sW.WriteLine("</tools>");
                sW.WriteLine("<directories>");
                sW.WriteLine("<build value=\"build\" />");
                sW.WriteLine("<include value=\"include\" />");
                sW.WriteLine("</directories>");
                sW.WriteLine("<sound adpcm=\"true\" />");
                sW.WriteLine("<motion version=\"3\" />");
                sW.WriteLine("<folders />");
            }

            // Para amabbs versiones...
            sW.WriteLine("</options>");
            sW.WriteLine("<set-default name=\"MEDIA\" value=\"media\" />");
            sW.WriteLine("<set name=\"SOUNDS\" value=\"${MEDIA}/sounds\" />");
            sW.WriteLine("<set name=\"MOTIONS\" value=\"${MEDIA}/motions\" />");
            sW.WriteLine("<set name=\"SCRIPTS\" value=\"${MEDIA}/scripts\" />");
            // Recursos
            sW.WriteLine("<resources>");
            sW.WriteLine("<!-- Sounds -->");
            insertarSonidosUPF(sW);
            sW.WriteLine("<!-- Motions -->");
            insertarMotions(sW);
            sW.WriteLine("<!-- Scripts -->");
            sW.WriteLine("<script path=\"sensors.p\" />");
            sW.WriteLine("</resources>");
            sW.WriteLine("</ugobe_project>");
            // Se cierra sW
            sW.Flush();
            sW.Close();
        }

        /// <summary>
        /// Crear el archivoP de la aplicación
        /// </summary>
        private void crearArchivoP()
        {
            FileStream fS = new FileStream(sRutaTemp + "\\sensors.p", FileMode.Create,
FileAccess.Write);
            StreamWriter sW = new StreamWriter(fS);

            crearCabeceraP(sW);
            crearCuerpoP(sW);
            sW.Flush();
            sW.Close();
        }

        /// <summary>
        /// Llama al compilador y le pasa los argumentos necesarios
        /// </summary>
        private bool llamarCompilador()
        {
            FileStream fS = new FileStream(sRutaTemp + "compile.bat", FileMode.Create,
FileAccess.Write);
            StreamWriter sW = new StreamWriter(fS);

            // crear bat
            sW.WriteLine("PATH %PATH%" + sRutaCompilador + "\\bin;");
            sW.WriteLine((Path.GetPathRoot(sRutaTemp)).Replace("\\",""));
            sW.WriteLine("cd " + sRutaTemp);
            sW.WriteLine("ugobe_project_tool " + sNombreProyecto + ".upf rebuild > " +
sRutaTemp + "resultadoCompilacion.txt");
            sW.WriteLine();
            sW.Flush();
            sW.Close();

            Process proc = new Process();
            proc.StartInfo.FileName = "compile.bat";
            proc.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
            proc.StartInfo.ErrorDialog = false;
```

```
            proc.StartInfo.WorkingDirectory = Path.GetDirectoryName(sRutaTemp);
            proc.Start();
            proc.WaitForExit();

            if (proc.ExitCode != 0)
            {
                return false;
            }

            swLog.WriteLine("\n----------TRAZA COMPILACION---------");
            FileStream fSR = new FileStream(sRutaTemp + "resultadoCompilacion.txt",
FileMode.Open, FileAccess.Read);
            StreamReader sR = new StreamReader(fSR);
            while (!sR.EndOfStream)
            {
                swLog.WriteLine(sR.ReadLine());
            }
            swLog.WriteLine("\n---------- FIN TRAZA COMPILACION---------");
            sR.Close();

            string sRutaURF = sRutaTemp + "\\build\\" + sNombreProyecto + ".urf";


            try
            {
                // Borrar el anterior archivo resultante si lo hubiera
                File.Delete(sRutaDestino + "\\" + sNombreProyecto + ".urf");

                // Cuando no exista el arhcivo, no se podra mover, por lo que ha habido
un error
                File.Copy(sRutaURF, sRutaDestino + "\\" + sNombreProyecto + ".urf");
            }

            catch (FileNotFoundException)
            {
                swLog.WriteLine("No se ha completado el compilado con éxito");
                return false;
            }
            catch (DirectoryNotFoundException)
            {
                swLog.WriteLine("No se ha completado el compilado con éxito. La ruta de
destino es errónea");
                return false;
            }

            swLog.WriteLine("\nCompilación realizada con éxito");
            return true;
        }

        /// <summary>
        /// Inserta en el archivo UPF los recursos de sonido necesarios
        /// </summary>
        /// <param name="sW">StreamWriter en el que escribir</param>
        private void insertarSonidosUPF(StreamWriter sW)
        {
            XmlDocument xDoc = new XmlDocument();
            xDoc.Load(sRutaXML);

            XmlNodeList personalidad = xDoc.GetElementsByTagName("Personalidad");
            XmlNodeList secuencias =
((XmlElement)personalidad[0]).GetElementsByTagName("Secuencia");

            foreach (XmlElement secuencia in secuencias)
            {
                XmlNodeList ordenes = secuencia.GetElementsByTagName("Orden");
                foreach (XmlElement orden in ordenes)
                {
                    XmlNodeList tipo = orden.GetElementsByTagName("Tipo");
                    string sTipo = tipo[0].InnerText;
                    if (sTipo.CompareTo("Reproducir Sonido") == 0)
                    {
                        XmlNodeList ruta = orden.GetElementsByTagName("Ruta");
                        string sRuta = ruta[0].InnerText;
                        string nombre = Path.GetFileName(sRuta);
                        string rutaDestino = sRutaTemp + "media\\sounds\\" + nombre;

                        // Si ya se ha agregado el recurso se salta
                        if (!File.Exists(rutaDestino))
```

```
                        {
                            File.Copy(sRuta, rutaDestino);
                            sW.WriteLine("<sound path=\"${SOUNDS}/" + nombre + "\" />");
                        }
                        bSonidos = true;
                    }
                }
            }

        }

        /// <summary>
        /// Inserta en el archivo UPF los recursos de animaciones
        /// </summary>
        /// <param name="sW">StreamWriter en el que escribir</param>
        private void insertarMotions(StreamWriter sW)
        {
            XmlDocument xDoc = new XmlDocument();
            xDoc.Load(sRutaXML);

            XmlNodeList personalidad = xDoc.GetElementsByTagName("Personalidad");
            XmlNodeList secuencias =
((XmlElement)personalidad[0]).GetElementsByTagName("Secuencia");

            foreach (XmlElement secuencia in secuencias)
            {
                XmlNodeList ordenes = secuencia.GetElementsByTagName("Orden");
                foreach (XmlElement orden in ordenes)
                {
                    XmlNodeList tipo = orden.GetElementsByTagName("Tipo");
                    string sTipo = tipo[0].InnerText;
                    if (sTipo.CompareTo("Motion") == 0)
                    {
                        XmlNodeList ruta = orden.GetElementsByTagName("Ruta");
                        string sRuta = ruta[0].InnerText;
                        string nombre = Path.GetFileName(sRuta);
                        string rutaDestino = sRutaTemp + "media\\motions\\" + nombre;

                        // Si ya se ha agregado el recurso se salta
                        if (!File.Exists(rutaDestino))
                        {
                            File.Copy(sRuta, rutaDestino);
                            sW.WriteLine("<motion path=\"${MOTIONS}/" + nombre + "\"
/>");
                        }
                        bMotions = true;
                    }
                }
            }
        }

        /// <summary>
        /// Crea la cabecera del archivo P
        /// </summary>
        private void crearCabeceraP(StreamWriter sW)
        {
            sW.WriteLine("#pragma pack 1");
            sW.WriteLine("#include <Log.inc>");
            sW.WriteLine("#include <Script.inc>");
            sW.WriteLine("#include <Sensor.inc>");
            sW.WriteLine("#include <Sound.inc>");
            sW.WriteLine("#include <Util.inc>");
            sW.WriteLine("#include <Property.inc> ");
            sW.WriteLine("#include <Motion.inc> ");
            sW.WriteLine("#include <Joint.inc> ");

            // Se añaden los recursos que se necesiten
            if (bSonidos)
            {
                sW.WriteLine("#include \"sounds.inc\" ");
            }
            if (bMotions)
            {
                sW.WriteLine("#include \"motions.inc\"");
            }

            sW.WriteLine("");
            sW.WriteLine("public init()");
```

```
            sW.WriteLine("{");
            insertarOrdenesInicioP(sW);
            sW.WriteLine("}");
        }

        /// <summary>
        /// Crea el cuerp del archivo P
        /// </summary>
        private void crearCuerpoP(StreamWriter sW)
        {
            sW.WriteLine("");
            sW.WriteLine("public on_sensor(time, sensor_name: sensor, value)");
            sW.WriteLine("{");
            sW.WriteLine("\tnew name[32];");
            sW.WriteLine("\tsensor_get_name(sensor, name);");
            sW.WriteLine("\tprintf(\"sensors:on_sensor(%d, %s, %d)\\n\", time, name,
value);");
            sW.WriteLine("\tswitch (sensor)");
            sW.WriteLine("\t{");
            insertarOrdenesCuerpoP(sW);
            sW.WriteLine("\t}");
            sW.WriteLine("\treturn true;");
            sW.WriteLine("}");
            sW.WriteLine("");
            // Método de espera
            if (bEspera)
            {
                sW.WriteLine("wait(value)");
                sW.WriteLine("{");
                sW.WriteLine("\tnew curTime=time(); ");
                sW.WriteLine("\twhile(time()-curTime<(value*1000)) ");
                sW.WriteLine("\t{ ");
                sW.WriteLine("\t}");
                sW.WriteLine("}");
            }
            // espera para que termine el sonido
            if (bSonidos)
            {
                sW.WriteLine("");
                sW.WriteLine("bool: wait_for_sound(sound_name: sound = 0)");
                sW.WriteLine("{ ");
                sW.WriteLine("\twhile (sound_is_playing(sound) == true) ");
                sW.WriteLine("\t{");
                sW.WriteLine("\t} ");
                sW.WriteLine("}");
                sW.WriteLine("");
            }
        }


        /// <summary>
        /// Inserta las ordenes que se deben ejecutar al encendido de Pleo
        /// </summary>
        /// <param name="sW">Archivo .p</param>
        private void insertarOrdenesInicioP(StreamWriter sW)
        {
            XmlDocument xDoc = new XmlDocument();
            xDoc.Load(sRutaXML);

            XmlNodeList personalidad = xDoc.GetElementsByTagName("Personalidad");
            XmlNodeList secuencias =
((XmlElement)personalidad[0]).GetElementsByTagName("Secuencia");

            foreach (XmlElement secuencia in secuencias)
            {
                XmlNodeList evento = secuencia.GetElementsByTagName("Evento");
                int iEvento = int.Parse(evento[0].InnerText);
                if (iEvento == Util.eventoStart)
                {
                    XmlNodeList ordenes = secuencia.GetElementsByTagName("Orden");

                    int iOrdenes = ordenes.Count;
                    // Array de ordenes, colocadas según el turno
                    XmlElement[] xOrdenes = new XmlElement[iOrdenes];

                    // Para cada una de las ordenes, se guardan en un array dependiendo
del turno
                    foreach (XmlElement orden in ordenes)
```

```
                    {
                        XmlNodeList tipo = orden.GetElementsByTagName("Tipo");
                        int iTurnoOrden =
int.Parse((orden.GetElementsByTagName("Turno"))[0].InnerText);
                        xOrdenes[iTurnoOrden - 1] = orden;
                    }

                    // Para cada una de las ordenes se escribe en el programa su trozo
de ejecucción
                    int iIter = 0;
                    while (iIter < iOrdenes)
                    {
                        XmlElement orden = xOrdenes[iIter++];
                        insertarOrden(sW, orden);
                    }

                    return;
                }
            }


        }

        /// <summary>
        /// Se insertan las órdenes con los eventos que las lanzan
        /// </summary>
        /// <param name="sW">Archivo .p</param>
        private void insertarOrdenesCuerpoP(StreamWriter sW)
        {
            XmlDocument xDoc = new XmlDocument();
            xDoc.Load(sRutaXML);

            XmlNodeList personalidad = xDoc.GetElementsByTagName("Personalidad");
            XmlNodeList secuencias =
((XmlElement)personalidad[0]).GetElementsByTagName("Secuencia");

            // Para cada uno de los eventos se crea un switch y se escriben las ordenes
a ejecutar
            foreach (XmlElement secuencia in secuencias)
            {
                XmlNodeList evento = secuencia.GetElementsByTagName("Evento");
                XmlNodeList ordenes = secuencia.GetElementsByTagName("Orden");

                // Se cuentan las ordenes de esta secuencia, si es =0, se pasa a la
siguiente secuencia
                int iOrdenes = ordenes.Count;
                if (iOrdenes == 0)
                {
                    continue;
                }

                int iEvento = int.Parse(evento[0].InnerText);
                string sEvento = Util.getNombreEventoCompiladorPleo(iEvento);
                string sComp = "value ==0";

                // Si no se trata del evento de inicio se escribe
                if (iEvento != Util.eventoStart)
                {
                    // Dependiendo del evento se disparará con un valor u otro
                    sComp = "value==0";
                    if (iEvento == Util.eventoPosado || iEvento == Util.eventoAlzado)
                    {
                        bAlzadoPosado = true;
                        continue;
                    }

                    else if (iEvento > Util.eventoStart && iEvento <
Util.eventoObstaculo)
                    {
                        sComp = "value==0";
                    }
                    else if (iEvento == Util.eventoObstaculo || iEvento ==
Util.eventoFilo || iEvento == Util.eventoAbuso
                        || iEvento == Util.eventoObjetoBoca)
                    {
                        sComp = "value==1";
                    }
                    else if (iEvento == Util.eventoAgitado)
```

```
                    {
                        sComp = "value > 75";
                    }
                    else
                    {
                        sComp = "true";
                    }

                    sW.WriteLine("\t\tcase " + sEvento + ":");
                    sW.WriteLine("\t\t\tif (" + sComp + ")");
                    sW.WriteLine("\t\t\t{");

                    // Array de ordenes, colocadas según el turno
                    XmlElement[] xOrdenes = new XmlElement[iOrdenes];

                    // Para cada una de las ordenes, se guardan en un array dependiendo
del turno
                    foreach (XmlElement orden in ordenes)
                    {
                        XmlNodeList tipo = orden.GetElementsByTagName("Tipo");
                        int iTurnoOrden =
int.Parse((orden.GetElementsByTagName("Turno"))[0].InnerText);
                        xOrdenes[iTurnoOrden - 1] = orden;
                    }

                    // Para cada una de las ordenes se escribe en el programa su trozo
de ejecucción
                    int iIter = 0;
                    while (iIter < iOrdenes)
                    {
                        XmlElement orden = xOrdenes[iIter++];
                        sW.Write("\t\t\t");
                        insertarOrden(sW, orden);
                    }
                    sW.WriteLine("\t\t\t}");
                }
            }
            if (bAlzadoPosado == true)
            {
                insertarCaseAlzadoPosado(sW, secuencias);
            }
        }

        /// <summary>
        /// Se inserta el case correspondiente a alzado y posado
        /// </summary>
        /// <param name="sW">StreamWriter donde se escribira la orden</param>
        /// <param name="secuencias">Secuencia de la orden</param>
        private void insertarCaseAlzadoPosado(StreamWriter sW, XmlNodeList secuencias)
        {
            sW.WriteLine("\t\tcase SENSOR_PICKED_UP:");
            bool bPrimera = true;

            // Para cada uno de los eventos se crea un switch y se escriben las ordenes
a ejecutar
            foreach (XmlElement secuencia in secuencias)
            {
                XmlNodeList evento = secuencia.GetElementsByTagName("Evento");
                XmlNodeList ordenes = secuencia.GetElementsByTagName("Orden");

                // Se cuentan las ordenes de esta secuencia, si es =0, se pasa a la
siguiente secuencia
                int iOrdenes = ordenes.Count;
                if (iOrdenes == 0)
                    continue;

                int iEvento = int.Parse(evento[0].InnerText);
                string sEvento = Util.getNombreEventoCompiladorPleo(iEvento);

                if (iEvento != Util.eventoAlzado && iEvento != Util.eventoPosado)
                {
                    continue;
                }

                if (iEvento == Util.eventoAlzado)
                {
                    if (bPrimera)
                    {
```

```
                        sW.WriteLine("\t\t\tif (value==1)");
                        bPrimera = false;
                    }
                    else
                    {
                        sW.WriteLine("\t\t\telse");
                    }
                }
                else
                {
                    if (bPrimera)
                    {
                        bPrimera = false;
                        sW.WriteLine("\t\t\tif (value==0)");
                    }
                    else
                    {
                        sW.WriteLine("\t\t\telse");
                    }
                }
                sW.WriteLine("\t\t\t{");

                // Array de ordenes, colocadas según el turno
                XmlElement[] xOrdenes = new XmlElement[iOrdenes];

                // Para cada una de las ordenes, se guardan en un array dependiendo del
turno
                foreach (XmlElement orden in ordenes)
                {
                    XmlNodeList tipo = orden.GetElementsByTagName("Tipo");
                    int iTurnoOrden =
int.Parse((orden.GetElementsByTagName("Turno"))[0].InnerText);
                    xOrdenes[iTurnoOrden - 1] = orden;
                }

                // Para cada una de las ordenes se escribe en el programa su trozo de
ejecucción
                int iIter = 0;
                while (iIter < iOrdenes)
                {
                    XmlElement orden = xOrdenes[iIter++];
                    sW.Write("\t\t\t");
                    insertarOrden(sW, orden);
                }
                sW.WriteLine("\t\t\t}");
            }
        }


        /// <summary>
        /// Escribe en el fichero la orden que se la pase
        /// </summary>
        /// <param name="sW">Archivo .p</param>
        /// <param name="orden">Orden a escribir en el archivo .p</param>
        private void insertarOrden(StreamWriter sW, XmlElement orden)
        {
            string sTipo = (orden.GetElementsByTagName("Tipo"))[0].InnerText;

            if (sTipo.CompareTo("Neutro") == 0)
            {
                sW.WriteLine("\tmonitor_exec(\"joint neutral\");");
            }

            else if (sTipo.CompareTo("Movimiento Único") == 0)
            {
                int iGradosFin =
int.Parse((orden.GetElementsByTagName("AnguloFin"))[0].InnerText);
                int iMiembro =
int.Parse((orden.GetElementsByTagName("Miembro"))[0].InnerText);


                sW.WriteLine("\tmonitor_exec(\"joint angle " + iMiembro + " " +
iGradosFin + " \");");
            }

            else if (sTipo.CompareTo("Movimiento Repetitivo") == 0)
            {
```

```
                int iGradosFin =
int.Parse((orden.GetElementsByTagName("AnguloFin"))[0].InnerText);
                int iMiembro =
int.Parse((orden.GetElementsByTagName("Miembro"))[0].InnerText);
                int iGradosInicio =
int.Parse((orden.GetElementsByTagName("AnguloInicio"))[0].InnerText);
                int iRepeticiones =
int.Parse((orden.GetElementsByTagName("Repeticiones"))[0].InnerText);


                sW.WriteLine("\tmonitor_exec(\"joint range " + iMiembro + " " +
iRepeticiones + " " + iGradosInicio + " " + iGradosFin + " \");");
            }

            else if (sTipo.CompareTo("Captura Imagen") == 0)
            {
                string sNombre = (orden.GetElementsByTagName("Nombre"))[0].InnerText;

                sW.WriteLine("\tmonitor_exec(\"camera capture " + sNombre + ".bmp bmp
new\");");
                // Espera de 7 segundos tras mandar la orden de captura
                sW.WriteLine("\twait(7);");
                bEspera = true;
            }

            else if (sTipo.CompareTo("Captura Sonido") == 0)
            {
                string sNombre = (orden.GetElementsByTagName("Nombre"))[0].InnerText;
                int iSegundos =
int.Parse((orden.GetElementsByTagName("Tiempo"))[0].InnerText);

                sW.WriteLine("\tmonitor_exec(\"audio both " + sNombre + ".wav " +
iSegundos + "\");");
            }

            else if (sTipo.CompareTo("Reproducir Sonido") == 0)
            {
                string sRuta = (orden.GetElementsByTagName("Ruta"))[0].InnerText;
                string sNombre = Path.GetFileNameWithoutExtension(sRuta);

                sW.WriteLine("\tsound_play(snd_" + sNombre + ");");
                sW.WriteLine("\twait_for_sound(snd_" + sNombre + ");");
            }

            else if (sTipo.CompareTo("Espera") == 0)
            {
                int iSegundos =
int.Parse((orden.GetElementsByTagName("Tiempo"))[0].InnerText);
                sW.WriteLine("\twait(" + iSegundos + ");");
                bEspera = true;
            }

            else if (sTipo.CompareTo("Motion") == 0)
            {
                string sRuta = (orden.GetElementsByTagName("Ruta"))[0].InnerText;
                string sNombre = Path.GetFileNameWithoutExtension(sRuta);

                sW.WriteLine("\tmotion_play(mot_" + sNombre + ");");
                sW.WriteLine("\twhile (motion_is_playing(mot_" + sNombre + "))");
                sW.WriteLine("\t{");
                sW.WriteLine("\t\tsleep;");
                sW.WriteLine("\t}");
            }
        }

        /// <summary>
        /// Realiza las operaciones necesarias para el compilado
        /// </summary>
        /// <returns>true si se realiza con éxito</returns>
        public override bool compilar()
        {
            bSonidos = false;
            bMotions = false;
            bEspera = false;
            bAlzadoPosado = false;

            bool bCorrecto = crearEsqueleto();
            // Si no se ha podido crear el arbol de directorios se devuelve false
```

```
            if (!bCorrecto)
                return false;
            try
            {
                crearArchivoUPF();
                crearArchivoP();
            }
            catch (Exception ex)
            {
                swLog.WriteLine("Error al crear los archivos de entrada para el
compilador de Pleo. Error: " + ex.Message);
                return false;
            }
            if (llamarCompilador())
            {
                return true;
            }
            return false;
        }

        /// <summary>
        /// Comprueba si la carpeta que se ha seleccionado contiene el compilador
necesario
        /// </summary>
        /// <returns>true si la ruta es válida</returns>
        public override bool rutaCompiladorValida()
        {
            if (!File.Exists(sRutaCompilador + "\\bin\\pawncc.exe"))
            {
                return false;
            }
            if (!File.Exists(sRutaCompilador + "\\bin\\pawncc32.exe"))
            {
                return false;
            }
            if (!File.Exists(sRutaCompilador + "\\bin\\pawncc33.exe"))
            {
                return false;
            }
            if (!File.Exists(sRutaCompilador + "\\bin\\pleocc.exe"))
            {
                return false;
            }
            return true;
        }

    }

}
```

13.4.3.3 WindowCompilador.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Windows.Forms;
using System.IO;
using System.Diagnostics;

namespace PleoProg.Compilador
{
    /// <summary>
    /// Clase que muestra una ventana con las opciones de compilación y para iniciar la
compilación
    /// </summary>
    public partial class WindowCompilador : Window
```

```
    {
        /// <summary>
        /// StreamWriter donde se escribirá el log
        /// </summary>
        private StreamWriter swLog;

        /// <summary>
        /// Constructor de la clase WindowCompilador
        /// </summary>
        /// <param name="sRutaXML">Ruta de entrada del XML que contiene la
Personalidad</param>
        /// <param name="sNombreProyecto">Nombre del proyecto actual</param>
        /// <param name="sRutaTemp">Ruta temporal</param>
        public WindowCompilador(string sRutaXML, string sNombreProyecto, string
sRutaTemp)
        {
            InitializeComponent();

            this.sRutaXML = sRutaXML;
            this.sNombreProyecto = sNombreProyecto;
            this.sRutaTemp = sRutaTemp;

            agregarVersionesPleo();

            this.textCompiladorRuta.Text = Properties.Settings.Default.sRutaDestino;
            this.textCompiladorRuta.ToolTip = textCompiladorRuta.Text;

            this.comboCompiladorVersion.SelectedIndex = 1;

        }

        /// <summary>
        /// Ruta del archivo XML resultante de la creación del diagrama
        /// </summary>
        private string sRutaXML
        {
            get;
            set;
        }

        /// <summary>
        /// Ruta del compilador de la forma de vida
        /// </summary>
        private string sRutaCompilador
        {
            get;
            set;
        }

        /// <summary>
        /// Nombre del proyecto que se está programando
        /// </summary>
        private string sNombreProyecto
        {
            get;
            set;
        }

        /// <summary>
        /// Ruta temporal donde se guardaran los archivos de la compilación
        /// </summary>
        private string sRutaTemp
        {
            get;
            set;
        }

        /// <summary>
        /// Selecciona una ruta destino donde ubicar el archivo resultante de la
compilación
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCompiladorFileRuta_Click(object sender, RoutedEventArgs e)
        {
            FolderBrowserDialog fBD = new FolderBrowserDialog();
            if (this.textCompiladorRuta.Text != "")
                fBD.SelectedPath = this.textCompiladorRuta.Text;
```

```
            fBD.Description = "Selecciona destino del programa";
            fBD.ShowDialog();
            if (fBD.SelectedPath == "")
                return;
            this.textCompiladorRuta.Text = fBD.SelectedPath;
            this.textCompiladorRuta.ToolTip = fBD.SelectedPath;
        }

        /// <summary>
        /// Cancela las operaciones y cierra la ventana
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCompiladorCancelar_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
            this.Close();
        }

        /// <summary>
        /// Cuando se cambie el tipo de vida se agregan las versiones pertinentes al
combobox de versión
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void comboCompiladorFormaVida_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            agregarVersionesPleo();
        }

        /// <summary>
        /// Se agregan las versiones de Pleo para las que se proporciona soporte
        /// </summary>
        private void agregarVersionesPleo()
        {
            if (comboCompiladorVersion == null)
                return;
            //Se eliminan todos los elementos de este combobox antes de añadir
            int cont = this.comboCompiladorVersion.Items.Count;
            int i = 0;
            while (i < cont)
            {
                this.comboCompiladorVersion.Items.RemoveAt(i++);
            }

            //Si se ha elegido la forma de vida Pleo se muestran las versiones
existentes
            if (this.comboCompiladorVersion != null &&
this.comboCompiladorFormaVida.SelectedIndex == 0)
            {
                this.comboCompiladorVersion.Items.Add("1.0");
                this.comboCompiladorVersion.Items.Add("1.1");
                comboCompiladorVersion.SelectedIndex = 0;
                if (Properties.Settings.Default.sCompiladorPleo.CompareTo("") != 0)
                {
                    this.textCompiladorPDK.Text =
Properties.Settings.Default.sCompiladorPleo;
                    this.textCompiladorPDK.ToolTip = this.textCompiladorPDK.Text;
                    this.sRutaCompilador = this.textCompiladorPDK.Text;
                }
            }
        }

        /// <summary>
        /// Se comprueban los campos, se compila con el compilador apropiado y se guarda
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCompiladorCompilar_Click(object sender, RoutedEventArgs e)
        {
            Compilador c;
            string sRutaDestino = textCompiladorRuta.Text;

            if (sRutaDestino == "")
            {
```

```
                System.Windows.MessageBox.Show(this, "Debe seleccionar una ruta donde
guardar el programa", "Compilar", MessageBoxButton.OK, MessageBoxImage.Exclamation);
                return;

            }
            if (sRutaCompilador == "")
            {
                if (comboCompiladorFormaVida.SelectedIndex == 0 &&
(System.Windows.MessageBox.Show(this, "No se ha seleccionado la ruta del compilador.
¿Desea visitar la página de Ugobe para su descarga?", "Ruta del compilador",
MessageBoxButton.YesNo, MessageBoxImage.Error)) == MessageBoxResult.Yes)
                {
                    Process.Start("http://www.pleoworld.com/downloads/pdk.aspx");

                }
                else
                {
                    System.Windows.MessageBox.Show(this, "Debe seleccionar la ubicación
del compilador se la forma de vida", "Compilar", MessageBoxButton.OK,
MessageBoxImage.Exclamation);
                    return;
                }
            }
            if (comboCompiladorFormaVida.SelectedIndex == 0)
            {
                int iVersion = int.Parse(comboCompiladorVersion.Text.Replace(".", ""));

                //Se crea un log para grabar el proceso de compilado y que pueda ser
salvado en caso de que se encuentre algún error
                FileStream fS = new FileStream(sRutaTemp + "log.txt", FileMode.Create,
FileAccess.Write);
                swLog = new StreamWriter(fS);

                c = new CompiladorPleo(iVersion, sRutaXML, sRutaDestino,
sRutaCompilador, sNombreProyecto, sRutaTemp, swLog);
                if (!c.rutaCompiladorValida())
                {
                    System.Windows.MessageBox.Show(this, "La ruta del compilador
seleccionada no es válida", "Compilar", MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }
                bool bCorrecto = c.compilar();

                if (!bCorrecto)
                {
                    swLog.Flush();
                    swLog.Close();

                    //Se muestra un error y se pregunta si se desea guardar el log
                    if ((System.Windows.MessageBox.Show(this, "No se ha podido realizar
la compilación satisfactoriamente. ¿Desea guardar el archivo de log para revisar los
errores?", "Error en la Compilación", MessageBoxButton.YesNo, MessageBoxImage.Error)) ==
MessageBoxResult.Yes)
                    {
                        Microsoft.Win32.SaveFileDialog sFD = new
Microsoft.Win32.SaveFileDialog();
                        sFD.FileName = "log.txt";
                        sFD.InitialDirectory = sRutaTemp;
                        sFD.Filter = "TXT File (*.txt) | *.txt";
                        sFD.DefaultExt = ".txt";
                        sFD.Title = "Guardar como...";
                        sFD.CheckPathExists = true;

                        if (sFD.ShowDialog() == true)
                        {
                            File.Delete(sFD.FileName);
                            File.Copy(sRutaTemp + "log.txt", sFD.FileName);
                        }
                    }
                }
                else
                {
                    System.Windows.MessageBox.Show(this, "La compilación se ha realizado
con éxito", "Compilar", MessageBoxButton.OK, MessageBoxImage.Information);
                    this.Close();
                }

            }
```

```
        }

        /// <summary>
        /// Al pulsar sobre el botón de selección de carpeta se muestra una ventana para
buscar y seleccionar
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCompiladorFilePDK_Click(object sender, RoutedEventArgs e)
        {
            FolderBrowserDialog fBD = new FolderBrowserDialog();
            if (this.textCompiladorPDK.Text != "")
                fBD.SelectedPath = this.textCompiladorPDK.Text;

            fBD.Description = "Selecciona la carpeta del compilador de la forma de
vida";
            fBD.ShowDialog();
            if (fBD.SelectedPath == "")
                return;
            this.textCompiladorPDK.Text = fBD.SelectedPath;
            this.textCompiladorPDK.ToolTip = fBD.SelectedPath;
            this.sRutaCompilador = this.textCompiladorPDK.Text;
            Properties.Settings.Default.sCompiladorPleo = fBD.SelectedPath;
            Properties.Settings.Default.Save();
        }

        /// <summary>
        /// Al cerrar se realiza limpieza de los archivos temporales
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs
e)
        {
            //Si no está cerrado se graba y cierra
            if (swLog != null)
            {
                try
                {
                    swLog.Flush();
                }
                catch (ObjectDisposedException)
                {
                }
                swLog.Close();
                if (Properties.Settings.Default.bGuardarLog == true)
                {
                    File.Delete(Properties.Settings.Default.sRutaLog);
                    File.Copy(sRutaTemp +
"log.txt",Properties.Settings.Default.sRutaLog);
                }
            }
            try
            {
                //Se eliminan todos los archivos temporales
                Directory.Delete(sRutaTemp, true);
            }
            catch (Exception)
            {
                //Si por cualquier razón no se puede borrar se continua.
            }
        }

    }
}
```

13.4.3.4 WindowCompilador.xaml

```
<Window x:Class="PleoProg.Compilador.WindowCompilador"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Compilador" Height="300" Width="400" WindowStartupLocation="CenterOwner"
ShowInTaskbar="True" ResizeMode="NoResize" WindowStyle="SingleBorderWindow"
Icon="/PleoProg;component/Recursos/IcoConf.png" Closing="Window_Closing">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="0.3*" />
            <ColumnDefinition Width="0.2*" />
            <ColumnDefinition Width="0.5*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
        </Grid.RowDefinitions>

        <Label Name="labelCompiladorFormaVida" Content="Forma de Vida:"
Margin="0,0,10,0" Grid.Row="0" Grid.Column="0" HorizontalAlignment="Right"
VerticalAlignment="Center"/>
        <ComboBox  Name="comboCompiladorFormaVida" Margin="10,0,0,0" Grid.Row="0"
Grid.Column="1" Grid.ColumnSpan="2" HorizontalAlignment="Left"
VerticalAlignment="Center" Width="100" Height="25"
SelectionChanged="comboCompiladorFormaVida_SelectionChanged">
            <ComboBoxItem Content="Pleo" IsSelected="True"/>
        </ComboBox>

        <Label Name="labelCompiladorPDK" Content="Compilador:" Margin="0,0,10,0"
Grid.Row="1" Grid.Column="0" HorizontalAlignment="Right" VerticalAlignment="Center"/>
        <TextBox Name="textCompiladorPDK" Margin="10,0,0,0" Grid.Row="1" Grid.Column="1"
Grid.ColumnSpan="2" HorizontalAlignment="Left" VerticalAlignment="Center" Width="200"
IsReadOnly="True" />
        <Button Name="buttonCompiladorFilePDK" Content="..." Margin="215,0,0,0"
Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2" HorizontalAlignment="Left"
VerticalAlignment="Center" Width="30" Click="buttonCompiladorFilePDK_Click" />


        <Label Name="labelCompiladorVersion" Content="Versión:" Margin="0,0,10,0"
Grid.Row="2" Grid.Column="0" HorizontalAlignment="Right" VerticalAlignment="Center"/>
        <ComboBox  Name="comboCompiladorVersion" Margin="10,0,0,0" Grid.Row="2"
Grid.Column="1" Grid.ColumnSpan="2" HorizontalAlignment="Left"
VerticalAlignment="Center" Width="100" Height="25" />

        <Label Name="labelCompiladorRuta" Content="Destino:" Margin="0,0,10,0"
Grid.Row="3" Grid.Column="0" HorizontalAlignment="Right" VerticalAlignment="Center"/>
        <TextBox Name="textCompiladorRuta" Margin="10,0,0,0" Grid.Row="3"
Grid.Column="1" Grid.ColumnSpan="2" HorizontalAlignment="Left"
VerticalAlignment="Center" Width="200" IsReadOnly="True" />
        <Button Name="buttonCompiladorFileRuta" Content="..." Margin="215,0,0,0"
Grid.Row="3" Grid.Column="1" Grid.ColumnSpan="2" HorizontalAlignment="Left"
VerticalAlignment="Center" Width="30" Click="buttonCompiladorFileRuta_Click" />

        <Button Name="buttonCompiladorCompilar" Content="Compilar" Margin="0,0,0,0"
Grid.Row="4" Grid.Column="0" Grid.ColumnSpan="2" HorizontalAlignment="Center"
VerticalAlignment="Center" Width="75" Height="28" Click="buttonCompiladorCompilar_Click"
/>
        <Button Name="buttonCompiladorCancelar" Content="Cancelar" Margin="0,0,0,0"
Grid.Row="4" Grid.Column="2"  HorizontalAlignment="Center" VerticalAlignment="Center"
Width="75" Height="28" Click="buttonCompiladorCancelar_Click" />
    </Grid>
</Window>
```

13.4.4 PleoProg.Interfaz

13.4.4.1 GestorDSL.cs

```
using System;
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Shapes;
using PleoProg.Datos;
using PleoProg.Estructuras;
using PleoProg.GestorPersonalidad;
using PleoProg.Interfaz.Controles;
using System.IO;
using System.Runtime.Serialization;
using System.Runtime.Serialization.Formatters.Binary;

namespace PleoProg.Interfaz
{
    /// <summary>
    /// Clase gestor encargada de gestionar el diagrama DSL
    /// </summary>
    public class GestorDSL
    {
        /// <summary>
        /// Constructor de la clase GestorDSL
        /// </summary>
        /// <param name="canvas">Canvas donde se dibuja</param>
        /// <param name="cmbEvActual">CopmboBox donde se ve y selecciona la secuencia
actual</param>
        public GestorDSL(Canvas canvas, ComboBox cmbEvActual)
        {
            this.canvas = canvas;
            this.personalidad = new Personalidad();
            this.cmbEvActual = cmbEvActual;
        }

        /// <summary>
        /// Lista con los controles de ordenes mostrados
        /// </summary>
        private LinkedList<ControlOrden> lnkLsOrdenes = new LinkedList<ControlOrden>();
        /// <summary>
        /// Lista con los controles de eventos mostrados
        /// </summary>
        private LinkedList<ControlEvento> lnkLsEventos = new
LinkedList<ControlEvento>();

        /// <summary>
        /// Canvas donde se realiza el dibujado
        /// </summary>
        public Canvas canvas
        {
            get;
            private set;
        }

        /// <summary>
        /// ComboBox que muestra una lista con las secuencias dibujadas
        /// </summary>
        public ComboBox cmbEvActual
        {
            get;
            private set;
        }

        /// <summary>
        /// Orden seleccionada
        /// </summary>
        public ControlOrden ordenSeleccionada
        {
            get;
            set;
        }
```

```
        /// <summary>
        /// Evento seleccionado
        /// </summary>
        public ControlEvento eventoSeleccionado
        {
            get;
            set;
        }

        /// <summary>
        /// Evento actual
        /// </summary>
        public ControlEvento eventoActual
        {
            get;
            set;
        }

        /// <summary>
        /// Personalidad que se esta representando y editando
        /// </summary>
        public Personalidad personalidad
        {
            get;
            set;
        }

        /// <summary>
        /// Abre una nueva personalidad
        /// </summary>
        /// <param name="sRuta">Ruta del archivo de la personalidad</param>
        public void abrirPersonalidad(string sRuta)
        {
            IFormatter formatter = new BinaryFormatter();
            Stream stream = new FileStream(sRuta, FileMode.Open, FileAccess.Read,
FileShare.Read);
            stream.Flush();
            personalidad = (Personalidad)formatter.Deserialize(stream);
            stream.Flush();
            stream.Close();
            dibujarPersonalidad();
        }

        /// <summary>
        /// Guarda la personalidad actual
        /// </summary>
        /// <param name="sRuta">Ruta del archivo a guardar</param>
        public void guardarPersonalidad(string sRuta)
        {
            IFormatter formatter = new BinaryFormatter();
            Stream stream = new FileStream(sRuta, FileMode.Create, FileAccess.Write,
FileShare.None);
            stream.Flush();
            formatter.Serialize(stream, personalidad);
            stream.Flush();
            stream.Close();
        }

        /// <summary>
        /// Dibuja una personalidad recien cargada
        /// </summary>
        public void dibujarPersonalidad()
        {
            // Se dibuja cada secuencia
            List<Secuencia> lsSecuencias = personalidad.getSecuencias();
            Point ptControl = new Point(0, 0);

            ptControl.X = Util.anchoControlEvento;
            ptControl.Y = Util.altoControlEvento;

            foreach (Secuencia sec in lsSecuencias)
            {
                // crear control,dibujar evento y linea siguiente
                ControlEvento cE = new ControlEvento(sec, this, ptControl);

                dibujarControl(ptControl, sec, sec.evento);
                eventoActual = cE;
```

```
                LinkedList<Orden> lnkLsOrdenes = sec.getOrdenes();
                // Se dibujan las ordenes en orden de turno
                int iContOrdenes = lnkLsOrdenes.Count;
                Orden[] arrOrdenes = new Orden[iContOrdenes];

                foreach (Orden o in lnkLsOrdenes)
                {
                    arrOrdenes[o.iTurno - 1] = o;
                }

                foreach (Orden o in arrOrdenes)
                {
                    ptControl.X += 25 + Util.anchoControlOrden;
                    Point ptPosicion = new Point(ptControl.X, ptControl.Y);


                    // crear control,dibujar orden y linea siguiente
                    if (o.GetType() == typeof(Neutro))
                    {
                        ptPosicion.Y += 23;
                    }
                    ControlOrden cO = new ControlOrden(sec, o, this, ptPosicion);
                    dibujarControl(ptPosicion, sec, o);
                }

                ptControl.X = Util.anchoControlEvento;
                ptControl.Y += 40 + Util.altoControlEvento;
            }

        }

        /// <summary>
        /// Dibuja el control apropiado
        /// </summary>
        /// <param name="ptDib">Punto de dibujado dentro del canvas</param>
        /// <param name="sec">Secuencia a la que pertenece el evento u orden</param>
        /// <param name="accion">evento u orden a dibujar</param>
        public void dibujarControl(Point ptDib, Secuencia sec, Object accion)
        {
            double dbAnchoControl;
            double dbAltoControl;

            if (accion.GetType() == typeof(Neutro))
            {
                dbAnchoControl = Util.anchoControlOrden;
                dbAltoControl = Util.altoControlOrden;
            }
            else
            {
                dbAnchoControl = Util.anchoControlEvento;
                dbAltoControl = Util.altoControlEvento;
            }

            // Posición del control centrado en el clic
            Double dbPosXControl = ptDib.X - (dbAnchoControl / 2);
            Double dbPosYControl = ptDib.Y - (dbAltoControl / 2);


            // Evita crear el control fuera de los límites del canvas
            dbPosXControl = dbPosXControl < 0 ? 0 : dbPosXControl;
            dbPosYControl = dbPosYControl < 0 ? 0 : dbPosYControl;
            dbPosXControl = dbPosXControl > canvas.Width ? canvas.Width : dbPosXControl;
            dbPosYControl = dbPosYControl > canvas.Height ? canvas.Height :
dbPosYControl;

            Point ptControl = new Point(dbPosXControl, dbPosYControl);

            if (accion.GetType() == typeof(Neutro) || accion.GetType() ==
typeof(Movimiento) || accion.GetType() == typeof(CapturaImagen) || accion.GetType() ==
typeof(CapturaSonido) || accion.GetType() == typeof(ReproducirSonido) ||
accion.GetType() == typeof(Espera) || accion.GetType() == typeof(Motion))
            {
                // Se dibuja el control
                ControlOrden cO = new ControlOrden(sec, (Orden)accion, this, ptControl);
                canvas.Children.Add(cO);
                Canvas.SetLeft(cO, dbPosXControl);
                Canvas.SetTop(cO, dbPosYControl);
```

```
                Orden orden = (Orden)accion;
                Orden ordenAnt = sec.buscarOrden(orden.iTurno - 1);
                // Si no hay una orden anterior, se dibuja la linea desde el evento
                if (ordenAnt == null)
                {
                    ControlEvento cE = buscarControlEvento(sec.evento);

                    Point ptOrigen = new Point(dbPosXControl, dbPosYControl);
                    Point ptDestino = new Point(Canvas.GetLeft(cE), Canvas.GetTop(cE));

                    // Se dibuja la flecha y se guarda
                    Line linea = nuevaLinea(ptOrigen, false, ptDestino, true);
                    cE.lineaSig = linea;
                    cO.lineaAnt = linea;

                }
                // Si hay una orden anterior, se une con la nueva a través de una línea
                else
                {
                    ControlOrden ant = buscarControlOrden(ordenAnt);

                    Point ptOrigen = new Point(dbPosXControl, dbPosYControl);
                    Point ptDestino = new Point(Canvas.GetLeft(ant),
Canvas.GetTop(ant));

                    // Se dibuja la flecha y se guarda
                    Line linea = nuevaLinea(ptOrigen, false, ptDestino, false);
                    ant.lineaSig = linea;
                    cO.lineaAnt = linea;
                }
                lnkLsOrdenes.AddLast(cO);
            }
            // Si se está anadiendo un evento se dibuja, no se dibujan lineas porque
siempre es el primer elemento a dibujar
            else
            {
                ControlEvento cE = new ControlEvento(sec, this, ptControl);
                canvas.Children.Add(cE);
                Canvas.SetLeft(cE, dbPosXControl);
                Canvas.SetTop(cE, dbPosYControl);
                lnkLsEventos.AddLast(cE);

                eventoActual = cE;
                añadirElementoComboBox(cE.evento.iId);
            }

        }

        /// <summary>
        /// Busca entre los controles de ordenes el que se corresponda con la orden que
se le pasa
        /// </summary>
        /// <param name="orden">Orden a buscar</param>
        /// <returns>ControlOrden que se corresponde con la orden</returns>
        public ControlOrden buscarControlOrden(Orden orden)
        {
            foreach (ControlOrden cO in lnkLsOrdenes)
                if (cO.orden.Equals(orden))
                    return cO;
            return null;
        }

        /// <summary>
        /// Busca entre los controles de eventos el que se corresponda con el evento que
se le pasa
        /// </summary>
        /// <param name="ev">Evento a buscar</param>
        /// <returns>ControlEvento que se corresponde con el evento</returns>
        public ControlEvento buscarControlEvento(Evento ev)
        {
            foreach (ControlEvento cE in lnkLsEventos)
                if (cE.evento.Equals(ev))
                    return cE;
            return null;
        }

        /// <summary>
```

```
        /// Redibuja una linea
        /// </summary>
        /// <param name="ptOrigen">Origen</param>
        /// <param name="bOrEv">true si se trata de una linea proveniente de un
evento</param>
        /// <param name="ptDestino">Destino</param>
        /// <param name="bDesEv">true si se trata de una linea que va hacia un
evento</param>
        /// <param name="linea">Linea</param>
        private void dibujarLinea(Point ptOrigen, bool bOrEv, Point ptDestino, bool
bDesEv, Line linea)
        {
            // Averiguar hacia que lado orientar la línea en el eje X
            if (ptOrigen.X > ptDestino.X)
            {
                if (bOrEv)
                    ptOrigen.X += 0;
                else
                    ptOrigen.X += 12;

                if (bDesEv)
                    ptDestino.X += 130;
                else
                    ptDestino.X += 162;
            }
            else
            {
                if (bOrEv)
                    ptOrigen.X += 130;
                else
                    ptOrigen.X += 162;

                if (bDesEv)
                    ptDestino.X += 0;
                else
                    ptDestino.X += 12;
            }
            if (bOrEv)
                ptOrigen.Y += 65;
            else
                ptOrigen.Y += 48;

            if (bDesEv)
                ptDestino.Y += 65;
            else
                ptDestino.Y += 48;

            linea.X1 = ptOrigen.X;
            linea.Y1 = ptOrigen.Y;

            linea.X2 = ptDestino.X;
            linea.Y2 = ptDestino.Y;
        }

        /// <summary>
        /// Cea una nueva linea entre el origen y el destino
        /// </summary>
        /// <param name="ptOrigen">Origen</param>
        /// <param name="bOrEv">true si se trata de un evento</param>
        /// <param name="ptDestino">Destino</param>
        /// <param name="bDesEv">true si el destino es un evento</param>
        /// <returns></returns>
        public Line nuevaLinea(Point ptOrigen, bool bOrEv, Point ptDestino, bool bDesEv)
        {
            Line linea = new Line();
            linea.Stroke = Brushes.Black;
            linea.StrokeThickness = 2;

            dibujarLinea(ptOrigen, bOrEv, ptDestino, bDesEv, linea);

            canvas.Children.Add(linea);
            return linea;
        }

        /// <summary>
        /// Redibuja la linea entre un control orden y un punto
        /// </summary>
        /// <param name="cO"></param>
```

```
        /// <param name="ptOrigen"></param>
        public void reDibujarLinea(ControlOrden cO, Point ptOrigen)
        {
            Orden sig = cO.secuencia.buscarOrden(cO.orden.iTurno + 1);
            Orden ant = cO.secuencia.buscarOrden(cO.orden.iTurno - 1);

            Line linea;
            Point ptDestino;

            if (ant != null)
            {
                ControlOrden cOAnt = buscarControlOrden(ant);
                ptDestino = new Point(Canvas.GetLeft(cOAnt), Canvas.GetTop(cOAnt));
                linea = cO.lineaAnt;

                canvas.Children.Remove(linea);
                dibujarLinea(ptOrigen, false, ptDestino, false, linea);
                canvas.Children.Add(linea);
            }

            // Si no hay orden anterior, hay un evento
            else
            {
                Secuencia sec = cO.secuencia;
                ControlEvento cE = buscarControlEvento(sec.evento);

                ptDestino = new Point(Canvas.GetLeft(cE), Canvas.GetTop(cE));
                linea = cO.lineaAnt;

                canvas.Children.Remove(linea);
                dibujarLinea(ptOrigen, false, ptDestino, true, linea);
                canvas.Children.Add(linea);

            }

            if (sig != null)
            {
                ControlOrden cOSig = buscarControlOrden(sig);
                ptDestino = new Point(Canvas.GetLeft(cO), Canvas.GetTop(cO));
                ptOrigen = new Point(Canvas.GetLeft(cOSig), Canvas.GetTop(cOSig));
                linea = cO.lineaSig;

                canvas.Children.Remove(linea);
                dibujarLinea(ptOrigen, false, ptDestino, false, linea);
                canvas.Children.Add(linea);
            }

        }

        /// <summary>
        /// Redibuja la línea entre un Control que representa un evento y otro punto
        /// </summary>
        /// <param name="cE">ControlEvento</param>
        /// <param name="ptOrigen">Punto de origen de la línea</param>
        public void reDibujarLinea(ControlEvento cE, Point ptOrigen)
        {
            Orden sig = cE.secuencia.buscarOrden(1);

            if (sig == null)
                return;
            else
            {
                ControlOrden cOSig = buscarControlOrden(sig);
                Line linea;
                Point ptDestino;

                if (sig != null)
                {
                    ptDestino = new Point(Canvas.GetLeft(cE), Canvas.GetTop(cE));
                    ptOrigen = new Point(Canvas.GetLeft(cOSig), Canvas.GetTop(cOSig));
                    linea = cE.lineaSig;

                    canvas.Children.Remove(linea);
                    dibujarLinea(ptOrigen, false, ptDestino, true, linea);
                    canvas.Children.Add(linea);
                }
            }
        }
```

```
        /// <summary>
        /// Incrementa el turno de la orden seleccionada
        /// </summary>
        /// <param name="cO"></param>
        public void incrementarTurno(ControlOrden cO)
        {
            // intercambiar el turno con el siguiente
            int iTurno = cO.orden.iTurno;
            Secuencia sec = cO.secuencia;
            Orden sig = sec.buscarOrden(iTurno + 1);

            if (sig != null)
            {
                ControlOrden cOSig = buscarControlOrden(sig);
                cO.orden.iTurno = sig.iTurno;
                cO.labelTurno.Content = cO.orden.iTurno;

                sig.iTurno = sig.iTurno - 1;
                cOSig.labelTurno.Content = sig.iTurno;

                // Modificación lineas de cada control
                Line aux = cO.lineaSig;
                cO.lineaSig = cOSig.lineaSig;
                cOSig.lineaAnt = cO.lineaAnt;
                cO.lineaAnt = aux;
                cOSig.lineaSig = aux;

                // Intercambio posiciones de los controles actual y siguiente
                Point ptPosSig = new Point(Canvas.GetLeft(cO), Canvas.GetTop(cO));
                Point ptPosCO = new Point(Canvas.GetLeft(cOSig), Canvas.GetTop(cOSig));
                canvas.Children.Remove(cO);
                canvas.Children.Remove(cOSig);

                canvas.Children.Add(cO);
                Canvas.SetLeft(cO, ptPosCO.X);
                Canvas.SetTop(cO, ptPosCO.Y);

                canvas.Children.Add(cOSig);
                Canvas.SetLeft(cOSig, ptPosSig.X);
                Canvas.SetTop(cOSig, ptPosSig.Y);
            }
        }

        /// <summary>
        /// Decrementa el turno de la orden seleccionada
        /// </summary>
        /// <param name="cO"></param>
        public void decrementarTurno(ControlOrden cO)
        {
            // intercambiar el turno con el anterior
            int iTurno = cO.orden.iTurno;
            Secuencia sec = cO.secuencia;
            Orden ant = sec.buscarOrden(iTurno - 1);

            if (ant != null)
            {
                ControlOrden cOAnt = buscarControlOrden(ant);
                cO.orden.iTurno = ant.iTurno;
                cO.labelTurno.Content = cO.orden.iTurno;

                ant.iTurno = ant.iTurno + 1;
                cOAnt.labelTurno.Content = ant.iTurno;

                // Modificación lineas de cada control
                Line aux = cO.lineaAnt;
                cOAnt.lineaSig = cO.lineaSig;
                cO.lineaAnt = cOAnt.lineaAnt;
                cO.lineaSig = aux;
                cOAnt.lineaAnt = aux;

                // Intercambio posiciones de los controles actual y siguiente
                Point ptPosAnt = new Point(Canvas.GetLeft(cO), Canvas.GetTop(cO));
                Point ptPosCO = new Point(Canvas.GetLeft(cOAnt), Canvas.GetTop(cOAnt));
                canvas.Children.Remove(cO);
                canvas.Children.Remove(cOAnt);

                canvas.Children.Add(cO);
```

```
                Canvas.SetLeft(cO, ptPosCO.X);
                Canvas.SetTop(cO, ptPosCO.Y);

                canvas.Children.Add(cOAnt);
                Canvas.SetLeft(cOAnt, ptPosAnt.X);
                Canvas.SetTop(cOAnt, ptPosAnt.Y);
            }
        }

        /// <summary>
        /// Elimina el control que se le pase de la interfaz. Redibuja las lineas que
unen los controles
        /// </summary>
        /// <param name="control"></param>
        public void eliminarControlInterfaz(object control)
        {
            //Si se va a eliminar un evento
            if (control.GetType() == typeof(ControlEvento))
            {
                // Se busca la secuencia y evento a eliminar
                ControlEvento cE = (ControlEvento)control;
                Secuencia sec = cE.secuencia;
                Evento e = cE.evento;
                cE = buscarControlEvento(e);

                int ordenEliminar = 1;
                int elementos = sec.getOrdenes().Count;
                if (elementos > 0)
                {
                    // Se eliminan todas las ordenes del diagrama
                    while (ordenEliminar <= elementos)
                    {
                        Orden o = sec.buscarOrden(ordenEliminar);
                        ControlOrden cO = buscarControlOrden(o);
                        canvas.Children.Remove(cO.lineaAnt);
                        canvas.Children.Remove(cO);
                        ordenEliminar++;
                        lnkLsOrdenes.Remove(cO);
                        sec.eliminarOrden(o);
                    }
                }
                // Se elimina el control evento
                canvas.Children.Remove(cE);
                lnkLsEventos.Remove(cE);

                eventoSeleccionado.deseleccionar();
                eventoSeleccionado = null;
                eventoActual = null;
                eliminarElementoComboBox(cE.evento.iId);
            }

            // Si se va a eliminar una orden
            if (control.GetType() == typeof(ControlOrden))
            {
                ControlOrden cO = (ControlOrden)control;
                // Se borra la linea anterior, siguiente y se enlaza anterior con
siguiente
                Orden o = cO.orden;
                Secuencia sec = cO.secuencia;
                ControlEvento cE = buscarControlEvento(sec.evento);

                Orden ant = sec.buscarOrden(o.iTurno - 1);
                Orden sig = sec.buscarOrden(o.iTurno + 1);

                ControlOrden cOAnt;
                ControlOrden cOSig;

                // Si detrás de la orden a borrar hay otra
                if (sig != null)
                {
                    // Si antes de la orden a borrar hay otra orden
                    if (ant != null)
                    {
                        cOAnt = buscarControlOrden(ant);
                        cOSig = buscarControlOrden(sig);

                        Point ptOrigen = new Point(Canvas.GetLeft(cOSig),
Canvas.GetTop(cOSig));
```

```
                        Point ptDestino = new Point(Canvas.GetLeft(cOAnt),
Canvas.GetTop(cOAnt));

                        Line linea = nuevaLinea(ptOrigen, false, ptDestino, false);
                        cOAnt.lineaSig = linea;
                        cOSig.lineaAnt = linea;

                    }
                    // Si no hay ninguna orden antes de la orden a borrar, hay un evento
                    else
                    {
                        cOSig = buscarControlOrden(sig);

                        Point ptOrigen = new Point(Canvas.GetLeft(cOSig),
Canvas.GetTop(cOSig));
                        Point ptDestino = new Point(Canvas.GetLeft(cE),
Canvas.GetTop(cE));

                        Line linea = nuevaLinea(ptOrigen, false, ptDestino, true);
                        cE.lineaSig = linea;
                        cOSig.lineaAnt = linea;
                    }

                    // Se cambia el turno de todas las ordenes siguientes
                    while (sig != null)
                    {
                        cOSig = buscarControlOrden(sig);

                        // Se modifica el turno de la orden
                        sig.iTurno = sig.iTurno - 1;
                        cOSig.labelTurno.Content = sig.iTurno;

                        sig = sec.buscarOrden(sig.iTurno + 2);
                    }

                }

                // Si detras de la orden a borrar no hay nada, es el último
                else
                {
                    // Si antes de la orden a borrar hay otra orden
                    if (ant != null)
                    {
                        cOAnt = buscarControlOrden(ant);
                        cOAnt.lineaSig = null;
                    }

                    // Si no hay ninguna orden antes de la orden a borrar, hay un evento
                    else
                    {
                        cE.lineaSig = null;
                    }
                }


                if (cO.lineaAnt != null)
                    canvas.Children.Remove(cO.lineaAnt);
                if (cO.lineaSig != null)
                    canvas.Children.Remove(cO.lineaSig);
                canvas.Children.Remove(cO);
                lnkLsOrdenes.Remove(cO);

                ordenSeleccionada.deseleccionar();
                ordenSeleccionada = null;
                eventoActual = cE;
                eventoSeleccionado = null;
            }

        }

        /// <summary>
        /// Añade al combobox un nuevo elemento
        /// </summary>
        /// <param name="iEv">id del evento a añadir</param>
        public void añadirElementoComboBox(int iEv)
        {
            string sNuevoEvento = Util.getNombreEvento(iEv);
```

```
            if (existeElementoComboBox(sNuevoEvento))
                return;

            if (sNuevoEvento != null)
                cmbEvActual.Items.Add(sNuevoEvento);

            seleccionarElementoComboBox(iEv);
        }

        /// <summary>
        /// Comprueba si existe el evento en el combobox
        /// </summary>
        /// <param name="sEvento">cadena de evento</param>
        /// <returns>true si existe</returns>
        private bool existeElementoComboBox(string sEvento)
        {
            return cmbEvActual.Items.Contains(sEvento);
        }

        /// <summary>
        /// Elimina del combobox el evento que se le pasa
        /// </summary>
        /// <param name="iEv">evento a eliminar</param>
        public void eliminarElementoComboBox(int iEv)
        {
            string sAntiguoEv = Util.getNombreEvento(iEv);
            cmbEvActual.Items.Remove(sAntiguoEv);
        }

        /// <summary>
        /// Selecciona en el combobox el elemento que se le pasa
        /// </summary>
        /// <param name="iEv">id del evento a seleccionar</param>
        public void seleccionarElementoComboBox(int iEv)
        {
            string sNuevoEv = Util.getNombreEvento(iEv);
            if (!existeElementoComboBox(sNuevoEv))
                return;
            cmbEvActual.SelectedItem = sNuevoEv;
        }

    }
}
```

13.4.4.2 WindowAcerca.xaml.cs y WindowAcerca.cs

Esta clase ha sido descargada tal como se comentan en el capítulo implementación. Ha sido modificada en alguna de sus partes para ajustarla al contexto actual. No se incluye el código por considerarse que no representa el trabajo del autor, pero de todas formas puede consultarse en el CD.

13.4.4.3 WindowCapturaSonido.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using PleoProg.Estructuras;

namespace PleoProg.Interfaz
{
    /// <summary>
    /// Clase que muestra una ventana con las opciones necesarias para editar una orden
de Captura Sonido
    /// </summary>
    public partial class WindowCapturaSonido : Window
    {
        /// <summary>
        /// Constructor de la clase WindowCapturaSonido
        /// </summary>
        /// <param name="o">Orden que se va a modificar o crear</param>
        public WindowCapturaSonido(Orden o)
        {
            InitializeComponent();
            this.cS = ((CapturaSonido)o);
            this.textBoxNomArchivoSonido.Text = cS.sNombreArchivo;
            this.textBoxNomTiempoSonido.Text = "" + cS.iSegundos;
            this.cS = cS;
            this.textBoxNomArchivoSonido.Focus();
        }

        /// <summary>
        /// Nombre del archivo
        /// </summary>
        private string sNombre
        {
            get;
            set;
        }

        /// <summary>
        /// Segundos de grabación
        /// </summary>
        private int iSegundos
        {
            get;
            set;
        }

        /// <summary>
        /// Orden de captura
        /// </summary>
        private CapturaSonido cS
        {
            get;
            set;
        }

        /// <summary>
        /// Al presionar Aceptar se verifica la entrada de datos y se guarda
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonAceptar_Click(object sender, RoutedEventArgs e)
        {
            sNombre = this.textBoxNomArchivoSonido.Text;
            sNombre = sNombre.Replace(" ", "");

            if (this.textBoxNomTiempoSonido.Text == "")
```

```
            {
                MessageBox.Show("Introduzca un tiempo de grabación", "Grabación de
Sonido", MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }

            iSegundos = int.Parse(this.textBoxNomTiempoSonido.Text);

            if (iSegundos <3)
            {
                MessageBox.Show("Introduzca un tiempo de grabación entero positivo mayor
que 2 segundos para la grabación", "Grabación de Sonido", MessageBoxButton.OK,
MessageBoxImage.Error);
                return;
            }

            if (sNombre == "")
            {
                MessageBox.Show("Introduzca un nombre para la grabación", "Grabación de
Sonido", MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }

            cS.sNombreArchivo = sNombre;
            cS.iSegundos = iSegundos;

            this.DialogResult = true;
            this.Close();
        }

        /// <summary>
        /// Al cancelar no se guarda nada
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCancelar_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
            this.Close();
        }

        /// <summary>
        /// No se permiten algunas caracteres en el campo nombre
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void textBoxNomArchivo_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.OemComma || e.Key == Key.OemPeriod)
                e.Handled = true;
            e.Handled = false;
        }

        /// <summary>
        /// Solo se permiten numeros en el campo tiempo
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void textBoxNomTiempoSonido_KeyDown(object sender, KeyEventArgs e)
        {
            if ((e.Key >= Key.D0 && e.Key <= Key.D9) || (e.Key >= Key.NumPad0 && e.Key
<= Key.NumPad9))
                e.Handled = false;
            else
                e.Handled = true;
        }
    }
}
```

13.4.4.4 WindowCapturaSonido.xaml

```
<Window x:Class="PleoProg.Interfaz.WindowCapturaSonido"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Grabación de Sonido" Height="200" Width="300"
WindowStartupLocation="CenterOwner" ShowInTaskbar="True" ResizeMode="NoResize"
WindowStyle="SingleBorderWindow" Icon="/PleoProg;component/Recursos/IcoConf.png">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
        </Grid.RowDefinitions>

        <Label Content="Nombre:" Height="28" HorizontalAlignment="Right"
Name="labelArchivoSonido" VerticalAlignment="Center" Grid.Row="0" Grid.Column="0"/>
        <TextBox  Height="23" Width="170" HorizontalAlignment="Left" Margin="12,0,0,0"
Name="textBoxNomArchivoSonido" VerticalAlignment="Center"  Grid.Column="1" Grid.Row="0"
Grid.ColumnSpan="3" IsEnabled="True" AllowDrop="False"
KeyDown="textBoxNomArchivo_KeyDown" />

        <Label Content="Tiempo:" Height="28" HorizontalAlignment="Right"
Name="labelTiempoSonido" VerticalAlignment="Center" Grid.Row="1" Grid.Column="0"/>
        <TextBox  Height="23" Width="44" HorizontalAlignment="Left" Margin="12,0,0,0"
Name="textBoxNomTiempoSonido" VerticalAlignment="Center"  Grid.Column="1" Grid.Row="1"
KeyDown="textBoxNomTiempoSonido_KeyDown" />

        <Button Content="Aceptar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonAceptarSonido" VerticalAlignment="Center" Grid.Column="0"
Grid.Row="2" Grid.ColumnSpan="2" Click="buttonAceptar_Click" IsEnabled="True"
IsDefault="True" />
        <Button Content="Cancelar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonCancelarSonido" VerticalAlignment="Center" Grid.Column="2"
Grid.Row="2" Grid.ColumnSpan="2" Click="buttonCancelar_Click"  IsEnabled="True"
IsCancel="True" />


    </Grid>
</Window>
```

13.4.4.5 WindowEvento.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using PleoProg.Estructuras;

namespace PleoProg.Interfaz
{
    /// <summary>
    /// Clase que muestra una ventana para crear o modificar un evento
    /// </summary>
    public partial class WindowEvento : Window
    {
        /// <summary>
        /// Constructor de la clase WindowEvento.
        /// </summary>
        /// <param name="ev">Evento que se va a modificar o crear</param>
        /// <param name="gestor">GestorDSL que gestiona el evento</param>
        public WindowEvento(Evento ev, GestorDSL gestor)
        {
            InitializeComponent();
            this.ev = ev;
            this.comboBoxEventos.SelectedIndex = ev.iId == -1 ? 0 : ev.iId;
            this.comboBoxEventos.Focus();
            this.gestor = gestor;
        }

        /// <summary>
        /// Evento que crea o modifica
        /// </summary>
        private Evento ev
        {
            get;
            set;
        }

        /// <summary>
        /// Gestor del DSL. Utilizado para comprobar que no se crean eventos ya
existentes
        /// </summary>
        private GestorDSL gestor
        {
            get;
            set;
        }

        /// <summary>
        /// No se realiza ningún cambio
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCancelarEvento_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
            this.Close();
        }

        /// <summary>
        /// Se guarda el evento.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonAceptarEvento_Click(object sender, RoutedEventArgs e)
        {
            int iNuevoEv = comboBoxEventos.SelectedIndex;
```

```
            // Si es el mismo evento, se sale sin modiciar nada
            if (iNuevoEv == ev.iId)
            {
                this.DialogResult = true;
                this.Close();
                return;
            }

            bool existe = gestor.personalidad.buscarSecuencia(new Evento(iNuevoEv)) ==
null ? false : true;
            // Si el tipo de evento está repetido mostrar un mensaje
            if (existe)
            {
                System.Windows.MessageBox.Show(this, "No es posible añadir un evento ya
existente", "Insertar un evento", MessageBoxButton.OK, MessageBoxImage.Exclamation);
                return;
            }
            // Modificar el evento
            ev.iId = iNuevoEv;
            if (ev.iId < 0 || ev.iId > 23)
                this.DialogResult = false;
            else
                this.DialogResult = true;
            this.Close();
        }
    }
}
```

13.4.4.6 WindowEvento.xaml

```
<Window x:Class="PleoProg.Interfaz.WindowEvento"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Evento" Height="200" Width="400" WindowStartupLocation="CenterOwner"
ShowInTaskbar="True" ResizeMode="NoResize" WindowStyle="SingleBorderWindow"
Icon="/PleoProg;component/Recursos/IcoConf.png">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
        </Grid.RowDefinitions>

        <Label Content="Evento:" Height="28" HorizontalAlignment="Right"
Margin="12,0,0,0" Name="labelEventoSel" VerticalAlignment="Center" Grid.Row="0"
Grid.Column="0"/>

        <ComboBox  Height="23" Width="210" HorizontalAlignment="Left" Margin="12,0,0,0"
Name="comboBoxEventos" VerticalAlignment="Center"  Grid.Row="0" Grid.Column="1"
Grid.ColumnSpan="2">
            <ComboBoxItem Content="Encendido" />
            <ComboBoxItem Content="Botón Wake Up" />
            <ComboBoxItem Content="Acariciar Cabeza" />
            <ComboBoxItem Content="Acariciar Mentón" />
            <ComboBoxItem Content="Acariciar Espalda" />
            <ComboBoxItem Content="Acariciar Cuello" />
            <ComboBoxItem Content="Acariciar Pata Delantera Derecha" />
            <ComboBoxItem Content="Acariciar Pata Delantera Izquierda" />
            <ComboBoxItem Content="Acariciar Pata Trasera Derecha" />
            <ComboBoxItem Content="Acariciar Pata Trasera Izquierda" />
            <ComboBoxItem Content="Interruptor Pata Delantera Derecha" />
            <ComboBoxItem Content="Interruptor Pata Delantera Izquierda" />
            <ComboBoxItem Content="Interruptor Pata Trasera Derecha" />
            <ComboBoxItem Content="Interruptor Pata Trasera Izquierda" />
            <ComboBoxItem Content="Obstáculo" />
            <ComboBoxItem Content="Filo" />
            <ComboBoxItem Content="Sonido" />
            <ComboBoxItem Content="Posado" />
            <ComboBoxItem Content="Agitado" />
            <ComboBoxItem Content="Abuso" />
```

```
            <ComboBoxItem Content="Alzado" />
            <ComboBoxItem Content="Objeto en la Boca" />
            <ComboBoxItem Content="Variación de Luz" />
            <ComboBoxItem Content="Otro Pleo" />
        </ComboBox>
        <Button Content="Aceptar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonAceptarEvento" VerticalAlignment="Center" Grid.Column="0"
Grid.Row="1" Grid.ColumnSpan="2" Click="buttonAceptarEvento_Click" IsDefault="True" />
        <Button Content="Cancelar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonCancelarEvento" VerticalAlignment="Center" Grid.Column="1"
Grid.Row="1" Grid.ColumnSpan="2" Click="buttonCancelarEvento_Click" IsCancel="True" />

    </Grid>
</Window>
```

13.4.4.7 WindowMovimiento.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using PleoProg.Estructuras;
using PleoProg.Datos;

namespace PleoProg.Interfaz
{
    /// <summary>
    /// Clase que muestra una ventana con las opciones necesarias para crear o modificar
una orden de movimiento
    /// </summary>
    public partial class WindowMovimiento : Window
    {
        /// <summary>
        /// Orden a crear
        /// </summary>
        public Orden orden
        {
            get;
            private set;
        }

        /// <summary>
        /// Abre la ventana para edición de una orden ya creada
        /// </summary>
        /// <param name="o"></param>
        public WindowMovimiento(Movimiento o)
        {
            InitializeComponent();

            textBoxPosInicial.Text = o.iGradosInicio.ToString();
            textBoxPosFinal.Text = o.iGradosFin.ToString();

            if (o.bRepetitivo)
            {
                textBoxRep.Text = o.iRepeticiones.ToString();
                RBRepetitivo.IsChecked = true;
                RBUnico.IsChecked = false;
            }
            else
            {
                textBoxRep.Text = "1";
                RBRepetitivo.IsChecked = false;
                RBUnico.IsChecked = true;
            }

            comboBoxMiembrosMov.SelectedIndex = o.iMiembro;
```

```
            orden = o;
            this.comboBoxMiembrosMov.Focus();
        }

        /// <summary>
        /// Solo se permiten numeros y guuión (para representar negativos) en el campo
posicion inicial
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void textBoxPos_KeyDown(object sender, KeyEventArgs e)
        {
            if ((e.Key >= Key.D0 && e.Key <= Key.D9) || (e.Key >= Key.NumPad0 && e.Key
<= Key.NumPad9) || (e.Key == Key.OemMinus))
                e.Handled = false;
            else
                e.Handled = true;
        }

        /// <summary>
        /// Solo se permiten numeros y guuión (para representar negativos) en el campo
posición final
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void textBoxPosfinal_KeyDown(object sender, KeyEventArgs e)
        {
            if ((e.Key >= Key.D0 && e.Key <= Key.D9) || (e.Key >= Key.NumPad0 && e.Key
<= Key.NumPad9) || (e.Key == Key.OemMinus))
                e.Handled = false;
            else
                e.Handled = true;
        }

        /// <summary>
        /// Solo se permiten números en el campo repeticiones
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void textBoxRep_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key >= Key.D0 && e.Key <= Key.D9 || e.Key >= Key.NumPad0 && e.Key <=
Key.NumPad9)
                e.Handled = false;
            else
                e.Handled = true;
        }

        /// <summary>
        /// Se validan los datos y se guarda el movimiento si procede
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonAceptarMovimiento_Click(object sender, RoutedEventArgs e)
        {
            int iMiembro = comboBoxMiembrosMov.SelectedIndex;
            textBoxPosInicial.Text = (textBoxPosInicial.Text).Replace(" ", "");
            textBoxPosFinal.Text = (textBoxPosFinal.Text).Replace(" ", "");
            textBoxRep.Text = (textBoxRep.Text).Replace(" ", "");

            //Si el miembro no está escogido o no se ha completado el campo posicion
final
            if (textBoxPosFinal.Text == "")
            {
                MessageBox.Show("Complete el campo posición final", "Insertar un
movimiento", MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }

            int iGradosFinal=0;
            int iGradosInicial=0;
            int iRepeticiones=0;

            try
            {
                iGradosFinal = int.Parse(textBoxPosFinal.Text);
            }
            catch (FormatException)
```

```
            {
                MessageBox.Show("El formato del campo ángulo final es incorrecto",
"Insertar un movimiento", MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }

            if (RBRepetitivo.IsChecked == true)
            {
                //Si el miembro no está escogido o no se ha completado el campo posicion
final
                if (textBoxPosInicial.Text == "" || textBoxRep.Text == "")
                {
                    MessageBox.Show("Complete los campos posición inicial y
repeticiones", "Insertar un movimiento", MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }

                try
                {
                    iGradosInicial = int.Parse(textBoxPosInicial.Text);
                    iRepeticiones = int.Parse(textBoxRep.Text);
                }
                catch (FormatException)
                {
                    MessageBox.Show("El formato de los datos introducidos no es
correcto", "Insertar un movimiento", MessageBoxButton.OK, MessageBoxImage.Error);
                    return;
                }
            }

            bool bAngValido = anguloValidoPleo(iMiembro, iGradosFinal,
(RBRepetitivo.IsChecked==true? true:false), iGradosInicial);
            if (!bAngValido)
                return;

            if (RBUnico.IsChecked == true)
            {
                Movimiento m = (Movimiento)orden;
                m.iGradosFin = iGradosFinal;
                m.iMiembro = iMiembro;
                m.bRepetitivo = false;
            }
            else
            {
                Movimiento m = (Movimiento)orden;
                m.iGradosInicio = iGradosInicial;
                m.iGradosFin = iGradosFinal;
                m.iRepeticiones = iRepeticiones;
                m.iMiembro = iMiembro;
                m.bRepetitivo = true;
            }

            this.DialogResult = true;
        }

        /// <summary>
        /// No se raliza ninguna operación si se apreta cancelar.
        /// Se cierra la ventana actual.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCancelarMovimiento_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
            this.Close();
        }

        /// <summary>
        /// Se ejecuta al elegir el tipo de orden de movimiento repetitivo
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void RBRepetitivo_Checked(object sender, RoutedEventArgs e)
        {
            this.textBoxRep.IsEnabled = true;
            this.textBoxPosInicial.IsEnabled = true;
        }
```

```
        /// <summary>
        /// Se ejecuta al elegir el tipo de orden de movimiento único
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void RBUnico_Checked(object sender, RoutedEventArgs e)
        {
            this.textBoxPosInicial.IsEnabled = false;
            this.textBoxPosInicial.Text = "0";
            this.textBoxRep.IsEnabled = false;
            this.textBoxRep.Text = "1";
        }

        /// <summary>
        /// Constata que los ángulos introducidos para el movimiento son correctos
        /// </summary>
        /// <param name="iMiembro"></param>
        /// <param name="iGradosFinal"></param>
        /// <param name="bRepetitivo"></param>
        /// <param name="iGradosInicial"></param>
        /// <returns></returns>
        private bool anguloValidoPleo(int iMiembro, int iGradosFinal, bool bRepetitivo,
int iGradosInicial)
        {
            switch (iMiembro)
            {
                //Cabeza entre -90 y 90
                case (Util.movOjos):

                //Cola
                case (Util.movColaHorizontal):
                case (Util.movColaVertical):
                    if (iGradosFinal > 90 || iGradosFinal < -90)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-90,90]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Cadera entre -45 y 45
                case Util.movCaderaDerecha:
                case Util.movCaderaIzquierda:
                    if (iGradosFinal > 45 || iGradosFinal < -45)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-45,45]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Codo entre 0 y 90
                case (Util.movCodoDerecho):
                case (Util.movCodoIzquierdo):
                    if (iGradosFinal > 90 || iGradosFinal < 0)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [0,90]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Cuello horiz. entre -65 y 65
                case (Util.movCuelloHorizontal):
                    if (iGradosFinal > 65 || iGradosFinal < -65)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-65,65]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Cuello vert. entre -75 y 75
                case (Util.movCuelloVertical):
                    if (iGradosFinal > 75 || iGradosFinal < -75)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-75,75]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
```

```
                        return false;
                    }
                    break;

                //Hombro entre -30 y 55
                case (Util.movHombroDerecho):
                case (Util.movHombroIzquierdo):
                    if (iGradosFinal > 55 || iGradosFinal < -30)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-30,55]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Rodilla entre -90 y 0
                case (Util.movRodillaDerecha):
                case (Util.movRodillaIzquierda):
                    if (iGradosFinal > 0 || iGradosFinal < -90)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-90,0]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Torso entre -15 y 15
                case (Util.movTorso):
                    if (iGradosFinal > 15 || iGradosFinal < -15)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-15,15]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;
            }

            if (!bRepetitivo)
                return true;

            switch (iMiembro)
            {
                //Cabeza entre -90 y 90
                case (Util.movOjos):

                //Cola
                case (Util.movColaHorizontal):
                case (Util.movColaVertical):
                    if (iGradosInicial > 90 || iGradosInicial < -90)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-90,90]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Cadera entre -45 y 45
                case Util.movCaderaDerecha:
                case Util.movCaderaIzquierda:
                    if (iGradosInicial > 45 || iGradosInicial < -45)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-45,45]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Codo entre 0 y 90
                case (Util.movCodoDerecho):
                case (Util.movCodoIzquierdo):
                    if (iGradosInicial > 90 || iGradosInicial < 0)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [0,90]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;
```

```
                //Cuello horiz. entre -65 y 65
                case (Util.movCuelloHorizontal):
                    if (iGradosInicial > 65 || iGradosInicial < -65)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-65,65]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Cuello vert. entre -75 y 75
                case (Util.movCuelloVertical):
                    if (iGradosInicial > 75 || iGradosInicial < -75)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-75,75]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Hombro entre -30 y 55
                case (Util.movHombroDerecho):
                case (Util.movHombroIzquierdo):
                    if (iGradosInicial > 55 || iGradosInicial < -30)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-30,55]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Rodilla entre -90 y 0
                case (Util.movRodillaDerecha):
                case (Util.movRodillaIzquierda):
                    if (iGradosInicial > 0 || iGradosInicial < -90)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-90,0]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;

                //Torso entre -15 y 15
                case (Util.movTorso):
                    if (iGradosInicial > 15 || iGradosInicial < -15)
                    {
                        MessageBox.Show("La posición introducida es incorrecta. Debe
estar en el rango [-15,15]", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        return false;
                    }
                    break;
            }

            return true;

        }
    }
}
```

13.4.4.8 WindowMovimiento.xaml

```
<Window x:Class="PleoProg.Interfaz.WindowMovimiento"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Establecer Movimiento" Height="300" Width="400"
WindowStartupLocation="CenterOwner" ShowInTaskbar="True" ResizeMode="NoResize"
WindowStyle="SingleBorderWindow" Icon="/PleoProg;component/Recursos/IcoConf.png">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
         </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
```

```
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
        </Grid.RowDefinitions>

        <Label Content="Tipo de Movimiento:" Height="28" HorizontalAlignment="Right"
Margin="12,0,0,0" Name="labelTipoMov" VerticalAlignment="Center" Grid.Row="0"
Grid.Column="0"/>
        <RadioButton Name="RBUnico" Content="Único" Margin="12,0,0,0"
GroupName="TipMovimiento" Grid.Row="0" Grid.Column="1"  HorizontalAlignment="Left"
VerticalAlignment="Center" IsChecked="False" Checked="RBUnico_Checked" />
        <RadioButton Name="RBRepetitivo" Content="Repetitivo" Margin="70,0,12,0"
GroupName="TipMovimiento" Grid.Row="0" Grid.Column="1"  HorizontalAlignment="Right"
VerticalAlignment="Center" IsChecked="False" Checked="RBRepetitivo_Checked" />

        <Label Content="Miembro:" Height="28" HorizontalAlignment="Right"
Margin="12,0,0,0" Name="labelMiembro" VerticalAlignment="Center" Grid.Row="1"
Grid.Column="0"/>
        <ComboBox  Height="23" Width="160" HorizontalAlignment="Left" Margin="12,0,0,0"
Name="comboBoxMiembrosMov" VerticalAlignment="Center"  Grid.Column="1" Grid.Row="1">
            <ComboBoxItem Content="Hombro Derecho" />
            <ComboBoxItem Content="Codo Derecho" />
            <ComboBoxItem Content="Hombro Izquierdo" />
            <ComboBoxItem Content="Codo Izquierdo" />
            <ComboBoxItem Content="Cadera Izquierda" />
            <ComboBoxItem Content="Rodilla Izquierda" />
            <ComboBoxItem Content="Cadera Derecha" />
            <ComboBoxItem Content="Rodilla Derecha" />
            <ComboBoxItem Content="Torso" />
            <ComboBoxItem Content="Cola (Horizontalmente)" />
            <ComboBoxItem Content="Cola (Verticalmente)" />
            <ComboBoxItem Content="Cuello (Horizontalmente)" />
            <ComboBoxItem Content="Cuello (Verticalmente)" />
            <ComboBoxItem Content="Ojos" />
        </ComboBox>

        <Label Content="Posición final:" Height="28" HorizontalAlignment="Right"
Margin="12,0,0,0" Name="labelPosicionFinal" VerticalAlignment="Center" Grid.Row="2"
Grid.Column="0"/>
        <TextBox  Height="23" Width="54" HorizontalAlignment="Left" Margin="12,0,0,0"
Name="textBoxPosFinal" VerticalAlignment="Center"  Grid.Column="1" Grid.Row="2"
IsEnabled="True" KeyDown="textBoxPos_KeyDown" AllowDrop="False" MaxLength="3" />

        <Label Content="Posición de inicio:" Height="28" HorizontalAlignment="Right"
Margin="12,0,0,0" Name="labelPosicionInicio" VerticalAlignment="Center" Grid.Row="3"
Grid.Column="0"/>
        <TextBox  Height="23" Width="54" HorizontalAlignment="Left" Margin="12,0,0,0"
Name="textBoxPosInicial" VerticalAlignment="Center"  Grid.Column="1" Grid.Row="3"
IsEnabled="False" KeyDown="textBoxPos_KeyDown" AllowDrop="False" MaxLength="3" />

        <Label Content="Repeticiones:" Height="28" HorizontalAlignment="Right"
Margin="12,0,0,0" Name="labelRep" VerticalAlignment="Center" Grid.Row="4"
Grid.Column="0"/>
        <TextBox  Height="23" Width="54" HorizontalAlignment="Left" Margin="12,0,0,0"
Name="textBoxRep" VerticalAlignment="Center"  Grid.Column="1" Grid.Row="4"
IsEnabled="False" KeyDown="textBoxRep_KeyDown" MaxLength="2" AllowDrop="False" />

        <Button Content="Aceptar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonAceptarMovimiento" VerticalAlignment="Center"
Grid.Column="0" Grid.Row="5" Click="buttonAceptarMovimiento_Click" IsDefault="True" />
        <Button Content="Cancelar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonCancelarMovimiento" VerticalAlignment="Center"
Grid.Column="1" Grid.Row="5" Click="buttonCancelarMovimiento_Click" IsCancel="True" />
    </Grid>
</Window>
```

13.4.4.9 WindowNombreArchivo.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;

namespace PleoProg.Interfaz
{
    /// <summary>
    /// Clase que muestra una ventana para introducir un nombre de archivo
    /// </summary>
    public partial class WindowNombreArchivo : Window
    {
        /// <summary>
        /// Constructor de la clase WindowNombreArchivo
        /// </summary>
        /// <param name="sNombre">Nombre del archivo que se va a modificar</param>
        public WindowNombreArchivo(string sNombre)
        {
            InitializeComponent();
            this.sNombre = sNombre;
            this.textBoxNomArchivo.Focus();
        }

        /// <summary>
        /// Nombre del archivo
        /// </summary>
        public string sNombre
        {
            get;
            set;
        }

        /// <summary>
        /// Al aceptar se valida el nombre del archivo
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonAceptar_Click(object sender, RoutedEventArgs e)
        {
            sNombre = textBoxNomArchivo.Text;
            sNombre = sNombre.Replace(" ", "");

            if (sNombre == "")
            {
                MessageBox.Show("Introduzca un nombre para el archivo", "Nuevo archivo",
MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }

            this.DialogResult = true;
            this.Close();
        }

        /// <summary>
        /// Si se cancela no se realiza ningún cambio
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCancelar_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
            this.Close();
        }

        /// <summary>
```

```
        /// No se permiten caracteres especiales en el campo nombre de archivo
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void textBoxNomArchivo_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.OemComma || e.Key == Key.OemPeriod)
                e.Handled = true;
            e.Handled = false;
        }
    }
}
```

13.4.4.10 WindowNombreArchivo.xaml

```
<Window x:Class="PleoProg.Interfaz.WindowNombreArchivo"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Nombre de Archivo..." Height="150" Width="300"
WindowStartupLocation="CenterOwner" ShowInTaskbar="True" ResizeMode="NoResize"
WindowStyle="SingleBorderWindow" Icon="/PleoProg;component/Recursos/IcoConf.png">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
        </Grid.RowDefinitions>

        <Label Content="Nombre:" Height="28" HorizontalAlignment="Right"
Name="labelArchivo" VerticalAlignment="Center" Grid.Row="0" Grid.Column="0"/>
        <TextBox  Height="23" Width="170" HorizontalAlignment="Left" Margin="12,0,0,0"
Name="textBoxNomArchivo" VerticalAlignment="Center"  Grid.Column="1" Grid.Row="0"
Grid.ColumnSpan="3" IsEnabled="True" AllowDrop="False"
KeyDown="textBoxNomArchivo_KeyDown" />
        <Button Content="Aceptar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonAceptar" VerticalAlignment="Center" Grid.Column="0"
Grid.Row="1" Grid.ColumnSpan="2" Click="buttonAceptar_Click" IsEnabled="True"
IsDefault="True" />
        <Button Content="Cancelar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonCancelar" VerticalAlignment="Center" Grid.Column="2"
Grid.Row="1" Grid.ColumnSpan="2" Click="buttonCancelar_Click"  IsEnabled="True"
IsCancel="True" />


    </Grid>
</Window>
```

13.4.4.11 WindowNombreArchivoOrden.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using PleoProg.Estructuras;

namespace PleoProg.Interfaz
{
    /// <summary>
    /// Clase que muestra una ventana para introducir el nombre con el que se guardará
el archivo resultante de ejecutar una orden
    /// </summary>
    public partial class WindowNombreArchivoOrden : Window
    {
        /// <summary>
        /// Constructor de la clase WindowNombreArchivoOrden
        /// </summary>
        /// <param name="o">Orden que se va a modificar o crear con un archivo
adjunto</param>
        public WindowNombreArchivoOrden(Orden o)
        {
            InitializeComponent();
            this.orden = o;
            if (o.GetType() == typeof(CapturaImagen))
            {
                this.sNombre=((CapturaImagen)o).sNombreArchivo;
            }
            if (o.GetType() == typeof(CapturaSonido))
            {
                this.sNombre = ((CapturaSonido)o).sNombreArchivo;
            }

            this.textBoxNomArchivo.Text = sNombre;
            this.textBoxNomArchivo.Focus();
        }

        /// <summary>
        /// Nombre del archivo a guardar
        /// </summary>
        private string sNombre
        {
            get;
            set;
        }

        /// <summary>
        /// Orden a modificar
        /// </summary>
        private Orden orden
        {
            get;
            set;
        }

        /// <summary>
        /// Al aceptar se validan los datos introducidos y se guarda
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonAceptar_Click(object sender, RoutedEventArgs e)
        {
            sNombre = textBoxNomArchivo.Text;
            sNombre = sNombre.Replace(" ","");

            if (sNombre == "")
```

```
            {
                MessageBox.Show("Introduzca un nombre para el archivo", "Nuevo archivo",
MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }

            if (orden.GetType() == typeof(CapturaImagen))
            {
                ((CapturaImagen)orden).sNombreArchivo = sNombre;
            }
            if (orden.GetType() == typeof(CapturaSonido))
            {
                ((CapturaSonido)orden).sNombreArchivo = sNombre;
            }

            this.DialogResult = true;
            this.Close();
        }

        /// <summary>
        /// Al cancelar no se realiza ningún cambio
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCancelar_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
            this.Close();
        }

        /// <summary>
        /// No se permiten introducir caracteres especiales en el nombre del archivo
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void textBoxNomArchivo_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key == Key.OemComma || e.Key == Key.OemPeriod)
                e.Handled = true;
            e.Handled = false;
        }
    }
}
```

13.4.4.12 WindowNombreArchivoOrden.xaml

```
<Window x:Class="PleoProg.Interfaz.WindowNombreArchivoOrden"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Nombre de Archivo..." WindowStartupLocation="CenterOwner"
ShowInTaskbar="True" ResizeMode="NoResize" WindowStyle="SingleBorderWindow" Height="150"
Width="300" Icon="/PleoProg;component/Recursos/IcoConf.png">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
        </Grid.RowDefinitions>

        <Label Content="Nombre:" Height="28" HorizontalAlignment="Right"
Name="labelArchivo" VerticalAlignment="Center" Grid.Row="0" Grid.Column="0"/>
        <TextBox  Height="23" Width="170" HorizontalAlignment="Left" Margin="12,0,0,0"
Name="textBoxNomArchivo" VerticalAlignment="Center"  Grid.Column="1" Grid.Row="0"
Grid.ColumnSpan="3" IsEnabled="True" AllowDrop="False"
KeyDown="textBoxNomArchivo_KeyDown" />
        <Button Content="Aceptar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonAceptar" VerticalAlignment="Center" Grid.Column="0"
Grid.Row="1" Grid.ColumnSpan="2" Click="buttonAceptar_Click" IsEnabled="True"
IsDefault="True" />
```

```
        <Button Content="Cancelar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonCancelar" VerticalAlignment="Center" Grid.Column="2"
Grid.Row="1" Grid.ColumnSpan="2" Click="buttonCancelar_Click"  IsEnabled="True"
IsCancel="True" />

    </Grid>
</Window>
```

13.4.4.13 WindowOpcionesArchivo.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Windows.Forms;
using System.IO;

namespace PleoProg.Interfaz
{
    /// <summary>
    /// Clase que muestra una ventana con la opción de crear o cargar una personalidad
    /// </summary>
    public partial class WindowOpcionesArchivo : Window
    {
        /// <summary>
        /// Constructor de la clase WindowOpcionesArchivo
        /// </summary>
        public WindowOpcionesArchivo()
        {
            InitializeComponent();
            sOpcion = "";
            this.textNombreNuevo.Focus();
        }

        /// <summary>
        /// Cadena con el nombre de archivo o ruta seleccionada
        /// </summary>
        public string sOpcion
        {
            get;
            set;
        }

         /// <summary>
        /// Al seleccionar un nuevo proyecto se muestran el cuadro de insercción de
nombre de proyecto y se oculta los otros
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void RBNuevo_Checked(object sender, RoutedEventArgs e)
        {
            if (textRutaCargar == null)
                return;

            this.textRutaCargar.IsEnabled = false;
            this.textRutaCargar.Visibility = Visibility.Hidden;
            this.buttonExplorar.IsEnabled = false;
            this.buttonExplorar.Visibility = Visibility.Hidden;
            this.labelRutaCargar.IsEnabled = false;
            this.labelRutaCargar.Visibility = Visibility.Hidden;

            this.textNombreNuevo.IsEnabled = true;
            this.textNombreNuevo.Visibility = Visibility.Visible;
            this.labelNombreProyecto.IsEnabled = true;
            this.labelNombreProyecto.Visibility = Visibility.Visible;

            this.textNombreNuevo.Focus();
```

```
        }

        /// <summary>
        /// Al seleccionar cargar un nuevo proyecto se muestra un cuadro para
seleccionar la ruta y se oculta todo lo demás
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void RBCargar_Checked(object sender, RoutedEventArgs e)
        {
            this.textRutaCargar.IsEnabled = true;
            this.textRutaCargar.Visibility = Visibility.Visible;
            this.buttonExplorar.IsEnabled = true;
            this.buttonExplorar.Visibility = Visibility.Visible;
            this.labelRutaCargar.IsEnabled = true;
            this.labelRutaCargar.Visibility = Visibility.Visible;

            this.textNombreNuevo.IsEnabled = false;
            this.textNombreNuevo.Visibility = Visibility.Hidden;
            this.labelNombreProyecto.IsEnabled = false;
            this.labelNombreProyecto.Visibility = Visibility.Hidden;

            this.textRutaCargar.Focus();
        }

        /// <summary>
        /// Al aceptar se comprueba si todo es correcto
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonAceptar_Click(object sender, RoutedEventArgs e)
        {
            if (RBCargar.IsChecked == true)
            {
                sOpcion = textRutaCargar.Text;
                //Si no se ha introducido una ruta correcta se muestra un mensaje de
error.
                if (!File.Exists(sOpcion) || (sOpcion).Replace(" ", "") == "")
                {
                    System.Windows.MessageBox.Show(this, "La ruta introducida no es
correcta", "Cargar Proyecto", MessageBoxButton.OK, MessageBoxImage.Exclamation);
                    return;
                }
            }
            else
            {
                sOpcion = (textNombreNuevo.Text);
                if ((sOpcion).Replace(" ", "") == "")
                {
                    System.Windows.MessageBox.Show(this, "El nombre introducido no es
correcto", "Nuevo Proyecto", MessageBoxButton.OK, MessageBoxImage.Exclamation);
                    return;
                }
            }

            this.DialogResult = true;
            this.Close();
        }

        /// <summary>
        /// Al salir se cierra la ventana
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonSalir_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
            this.Close();
        }

        /// <summary>
        /// Se muestra una ventana para seleccionar el archivo que se desee guardar
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonExplorar_Click(object sender, RoutedEventArgs e)
        {
            OpenFileDialog oFD = new OpenFileDialog();
```

```
            oFD.InitialDirectory =
System.IO.Path.GetPathRoot(System.IO.Path.GetTempPath());
            oFD.Filter = "BIN File (*.bin) | *.bin";
            oFD.FilterIndex = 0;
            oFD.RestoreDirectory = true;
            oFD.ShowDialog();

            if (oFD.FileName != "")
            {
                sOpcion = oFD.FileName;
                this.textRutaCargar.Text = oFD.FileName;
                this.textRutaCargar.ToolTip = oFD.FileName;
            }
            else
                return;
        }
    }
}
```

13.4.4.14 WindowOpcionesArchivo.xaml

```
<Window x:Class="PleoProg.Interfaz.WindowOpcionesArchivo"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Opciones" Height="150" Width="400" WindowStartupLocation="CenterOwner"
ShowInTaskbar="True" ResizeMode="NoResize" WindowStyle="SingleBorderWindow"
Icon="/PleoProg;component/Recursos/IcoConf.png">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="0.3*" />
            <ColumnDefinition Width="0.3*" />
            <ColumnDefinition Width="0.3*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
        </Grid.RowDefinitions>
        <RadioButton Name="RBNuevo" Content="Nuevo Proyecto"  GroupName="Opcion"
Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="2"  Margin="0,0,0,0"
HorizontalAlignment="Center" VerticalAlignment="Center" IsChecked="True"
Checked="RBNuevo_Checked" />
        <TextBox Name="textNombreNuevo" Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2"
Height="23" Width="150" Margin="10,0,0,0" HorizontalAlignment="Left"
VerticalAlignment="Center" />

        <Label Name="labelNombreProyecto" Content="Nombre:" Grid.Row="1" Grid.Column="0"
Margin="0,0,10,0" HorizontalAlignment="Right" VerticalAlignment="Center" />
        <Label Name="labelRutaCargar" Content="Ruta:" Grid.Row="1" Grid.Column="0"
Margin="0,0,10,0" HorizontalAlignment="Right" VerticalAlignment="Center"
Visibility="Hidden" IsEnabled="False"/>

        <RadioButton Name="RBCargar" Content="Cargar Proyecto" GroupName="Opcion"
Grid.Row="0" Grid.Column="1"  Grid.ColumnSpan="2" Margin="0,0,0,0"
HorizontalAlignment="Center" VerticalAlignment="Center" IsChecked="false"
Checked="RBCargar_Checked" />
        <TextBox Name="textRutaCargar" Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2"
Height="23" Width="150" Margin="10,0,0,0" HorizontalAlignment="Left"
VerticalAlignment="Center" IsEnabled="False" Visibility="Hidden" IsReadOnly="True" />
        <Button Name="buttonExplorar" Content="..." Grid.Row="1" Grid.Column="1"
Grid.ColumnSpan="2" Height="23" Width="30" Margin="170,0,0,0" HorizontalAlignment="Left"
VerticalAlignment="Center" IsEnabled="False" Visibility="Hidden"
Click="buttonExplorar_Click" />

        <Button Name="buttonOpcionesAceptar" Content="Aceptar" Margin="0,0,0,0"
Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="2" HorizontalAlignment="Center"
VerticalAlignment="Center"  Width="75" Height="28" Click="buttonAceptar_Click"
IsDefault="True" />
        <Button Name="buttonOpcionesSalir" Content="Salir" Margin="0,0,0,0" Grid.Row="2"
Grid.Column="1" Grid.ColumnSpan="2" HorizontalAlignment="Center"
VerticalAlignment="Center" Width="75" Height="28" Click="buttonSalir_Click"
IsCancel="True" />
    </Grid>
</Window>
```

13.4.4.15 WindowPreferencias.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Windows.Forms;

namespace PleoProg.Interfaz
{
    /// <summary>
    /// Clase que muestra una ventana para editar preferencias del programa
    /// </summary>
    public partial class WindowPreferencias : Window
    {
        /// <summary>
        /// Constructor de la clase WindowPreferencias
        /// </summary>
        public WindowPreferencias()
        {
            InitializeComponent();
            bool bSalvarLog = Properties.Settings.Default.bGuardarLog;
            string sRutaLog = Properties.Settings.Default.sRutaLog;

            string sRutaPDK = Properties.Settings.Default.sCompiladorPleo;
            string sRutaDestino = Properties.Settings.Default.sRutaDestino;

            if (bSalvarLog==true)
            {
                this.CBGuardarLog.IsChecked = true;
                this.textBoxRutaLog.Text = sRutaLog;
                this.textBoxRutaLog.ToolTip = sRutaLog;
            }

            this.textBoxRutaCompiladorPleo.Text = sRutaPDK;
            this.textBoxRutaCompiladorPleo.ToolTip = sRutaPDK;

            this.textBoxRutaDestino.Text = sRutaDestino;
            this.textBoxRutaDestino.ToolTip = sRutaDestino;

        }

        /// <summary>
        /// Si se desea guardar (o no) el log se activan (o desactiva) los campos
pertinentes
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void CBGuardarLog_Checked(object sender, RoutedEventArgs e)
        {
            if (CBGuardarLog.IsChecked == true)
            {
                this.labelRutaLog.IsEnabled = true;
                this.textBoxRutaLog.IsEnabled = true;
                this.buttonExplorar.IsEnabled = true;
            }
            else
            {
                this.labelRutaLog.IsEnabled = true;
                this.textBoxRutaLog.IsEnabled = true;
                this.buttonExplorar.IsEnabled = true;
            }
        }

        /// <summary>
        /// Permite seleccionar la ruta del log
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
```

```
        private void buttonExplorar_Click(object sender, RoutedEventArgs e)
        {
            Microsoft.Win32.SaveFileDialog sFD = new Microsoft.Win32.SaveFileDialog();
            sFD.FileName = "log.txt";

            if (Properties.Settings.Default.sRutaLog == "")
                sFD.InitialDirectory =
System.IO.Path.GetPathRoot(System.IO.Path.GetTempPath());
            else
                sFD.InitialDirectory = Properties.Settings.Default.sRutaLog;

            sFD.Filter = "TXT File (*.txt) | *.txt";
            sFD.DefaultExt = ".txt";
            sFD.Title = "Guardar como...";
            sFD.CheckPathExists = true;

            if (sFD.ShowDialog() == true)
            {
                this.textBoxRutaLog.Text = sFD.FileName;
                this.textBoxRutaLog.ToolTip = sFD.FileName;
                Properties.Settings.Default.sRutaLog = sFD.FileName;
            }
        }

        /// <summary>
        /// Permite seleccionar la ruta del compilador
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonExplorarCompilador_Click(object sender, RoutedEventArgs e)
        {
            FolderBrowserDialog fBD = new FolderBrowserDialog();
            if (Properties.Settings.Default.sCompiladorPleo!="")
                fBD.SelectedPath = Properties.Settings.Default.sCompiladorPleo;

            fBD.Description = "Selecciona la carpeta del compilador de la forma de
vida";
            fBD.ShowDialog();
            if (fBD.SelectedPath == "")
                return;
            this.textBoxRutaCompiladorPleo.Text = fBD.SelectedPath;
            this.textBoxRutaCompiladorPleo.ToolTip = fBD.SelectedPath;

            Properties.Settings.Default.sCompiladorPleo = fBD.SelectedPath;
        }

        /// <summary>
        /// Guarda las preferencias seleccionadas
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonAceptar_Click(object sender, RoutedEventArgs e)
        {
            if (CBGuardarLog.IsChecked == true)
                Properties.Settings.Default.bGuardarLog = true;
            else
                Properties.Settings.Default.bGuardarLog = false;

            Properties.Settings.Default.Save();
            this.Close();
        }

        /// <summary>
        /// Permite seleccionar la ruta de guardar el programa por defecto
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonExplorarDestino_Click(object sender, RoutedEventArgs e)
        {
            FolderBrowserDialog fBD = new FolderBrowserDialog();
            if (Properties.Settings.Default.sRutaDestino!= "")
                fBD.SelectedPath = Properties.Settings.Default.sRutaDestino;

            fBD.Description = "Selecciona destino del programa";
            fBD.ShowDialog();
            if (fBD.SelectedPath == "")
                return;
            this.textBoxRutaDestino.Text = fBD.SelectedPath;
```

```
            this.textBoxRutaDestino.ToolTip = fBD.SelectedPath;

            Properties.Settings.Default.sRutaDestino = this.textBoxRutaDestino.Text;
            Properties.Settings.Default.Save();
        }

        /// <summary>
        /// Sale y no guarda nada
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCancelar_Click(object sender, RoutedEventArgs e)
        {
            Properties.Settings.Default.Reload();
            Properties.Settings.Default.Save();
            this.Close();
        }
    }
}
```

13.4.4.16 WindowPreferencias.xaml

```
<Window x:Class="PleoProg.Interfaz.WindowPreferencias"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Preferencias" Height="275" Width="400"
WindowStartupLocation="CenterOwner" ShowInTaskbar="True" ResizeMode="NoResize"
WindowStyle="SingleBorderWindow" Icon="/PleoProg;component/Recursos/IcoConf.png">
    <Grid>
        <Grid.ColumnDefinitions>
        <ColumnDefinition Width="1*" />
        <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />

        </Grid.RowDefinitions>

        <CheckBox Name="CBGuardarLog" Grid.Row="0" Grid.Column="0"
VerticalAlignment="Center" HorizontalAlignment="Right" Margin="0,0,90,0"
IsChecked="false" Checked="CBGuardarLog_Checked" />
        <Label Content="Guardar LOG" Grid.Row="0" Grid.Column="0"
VerticalAlignment="Center" HorizontalAlignment="Right"
               Margin="0,0,10,0"/>

        <Label Name="labelRutaLog" Content="Ruta del LOG" Grid.Row="1" Grid.Column="0"
VerticalAlignment="Center" HorizontalAlignment="Right" Margin="0,0,10,0"
IsEnabled="False"/>
        <TextBox Name="textBoxRutaLog" Grid.Row="1" Grid.Column="1" Grid.ColumnSpan="2"
Height="23" Width="170" VerticalAlignment="Center" HorizontalAlignment="Left"
Margin="10,0,0,0" IsEnabled="False" IsReadOnly="True" />
        <Button Name="buttonExplorar" Content="..." Grid.Row="1" Grid.Column="1"
Grid.ColumnSpan="2" Height="23" Width="30" Margin="190,0,0,0" HorizontalAlignment="Left"
VerticalAlignment="Center" IsEnabled="False" Click="buttonExplorar_Click" />

        <Label Name="labelRutaCompiladorPleo" Content="Ruta del PDK" Grid.Row="2"
Grid.Column="0" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="0,0,10,0"
ToolTip="Pleo Development Kit"/>
        <TextBox Name="textBoxRutaCompiladorPleo" Grid.Row="2" Grid.Column="1"
Grid.ColumnSpan="2" Height="23" Width="170" VerticalAlignment="Center"
HorizontalAlignment="Left" Margin="10,0,0,0" IsReadOnly="True" />
        <Button Name="buttonExplorarCompilador" Content="..." Grid.Row="2"
Grid.Column="1" Grid.ColumnSpan="2" Height="23" Width="30" Margin="190,0,0,0"
HorizontalAlignment="Left" VerticalAlignment="Center"
Click="buttonExplorarCompilador_Click" />

        <Label Name="labelRutaDestino" Content="Ruta destino" Grid.Row="3"
Grid.Column="0" VerticalAlignment="Center" HorizontalAlignment="Right" Margin="0,0,10,0"
ToolTip="Pleo Development Kit"/>
```

```
        <TextBox Name="textBoxRutaDestino" Grid.Row="3" Grid.Column="1"
Grid.ColumnSpan="2" Height="23" Width="170" VerticalAlignment="Center"
HorizontalAlignment="Left" Margin="10,0,0,0" IsReadOnly="True" />
        <Button Name="buttonExplorarDestino" Content="..." Grid.Row="3" Grid.Column="1"
Grid.ColumnSpan="2" Height="23" Width="30" Margin="190,0,0,0" HorizontalAlignment="Left"
VerticalAlignment="Center" Click="buttonExplorarDestino_Click" />
        <Button Content="Aceptar" Height="28" Width="75" HorizontalAlignment="Center"
Name="buttonAceptar" VerticalAlignment="Center" Grid.Column="0" Grid.Row="4"
Grid.ColumnSpan="2" Click="buttonAceptar_Click" IsEnabled="True" IsDefault="True" />
        <Button Content="Cancelar" Height="28" Width="75" HorizontalAlignment="Center"
Name="buttonCancelar" VerticalAlignment="Center" Grid.Column="1" Grid.Row="4"
Grid.ColumnSpan="2" Click="buttonCancelar_Click"  IsEnabled="True" IsCancel="True" />

    </Grid>
</Window>
```

13.4.4.17 WindowTiempoEspera.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using PleoProg.Estructuras;

namespace PleoProg.Interfaz
{
    /// <summary>
    /// Clase que muestra una ventana para modificar una orden de Espera
    /// </summary>
    public partial class WindowTiempoEspera : Window
    {
        /// <summary>
        /// Constructor de la clase WindowTiempoEspera
        /// </summary>
        /// <param name="eEsp">Orden de Espera a modificar</param>
        public WindowTiempoEspera(Espera eEsp)
        {
            InitializeComponent();
            this.eEsp = eEsp;
            this.textBoxTiempoEspera.Text =""+ eEsp.iEspera;
            this.textBoxTiempoEspera.Focus();
        }
        /// <summary>
        /// Orden de espera a crear o modificar
        /// </summary>
        public Espera eEsp
        {
            get;
            set;
        }

        /// <summary>
        /// Al aceptar se validan los datos y se guarda
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonAceptar_Click(object sender, RoutedEventArgs e)
        {
            int iTiempo = 0;
            try
            {
                iTiempo = int.Parse(textBoxTiempoEspera.Text);
            }
            catch (FormatException)
            {
```

```
                MessageBox.Show("El formato del tiempo introducido no es correco.
Introduzca los segundos", "Insertar tiempo de espera", MessageBoxButton.OK,
MessageBoxImage.Error);
                return;
            }
            if (iTiempo <= 0)
            {
                MessageBox.Show("El tiempo de espera debe ser un entero positivo mayor
de 0", "Insertar tiempo de espera", MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }

            eEsp.iEspera = iTiempo;
            this.DialogResult = true;
            this.Close();
        }
        /// <summary>
        /// Al cancelar se sale sin guardar
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCancelar_Click(object sender, RoutedEventArgs e)
        {
            this.DialogResult = false;
            this.Close();
        }

        /// <summary>
        /// Solo se permiten meter números en el campo espera
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void textBoxTiempoEspera_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.Key >= Key.D0 && e.Key <= Key.D9 || e.Key >= Key.NumPad0 && e.Key <=
Key.NumPad9)
                e.Handled = false;
            else
                e.Handled = true;
        }
    }
}
```

13.4.4.18 WindowTiempoEspera.xaml

```
<Window x:Class="PleoProg.Interfaz.WindowTiempoEspera"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Tiempo de Espera" Height="150" Width="300" ShowInTaskbar="True"
ResizeMode="NoResize" WindowStyle="SingleBorderWindow"
Icon="/PleoProg;component/Recursos/IcoConf.png" Name="TiempoEspera">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="1*" />
            <RowDefinition Height="1*" />
        </Grid.RowDefinitions>
        <Label Content="Tiempo (s):" Height="28" Margin="0,0,10,0"
HorizontalAlignment="Right"  Name="labelTiempo" VerticalAlignment="Center" Grid.Row="0"
Grid.Column="0"/>
        <TextBox  Height="23" Width="40" HorizontalAlignment="Left" Margin="12,0,0,0"
Name="textBoxTiempoEspera" VerticalAlignment="Center"  Grid.Column="1" Grid.Row="0"
IsEnabled="True" AllowDrop="False" KeyDown="textBoxTiempoEspera_KeyDown" MaxLength="2"
/>
        <Button Content="Aceptar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonAceptar" VerticalAlignment="Center" Grid.Column="0"
Grid.Row="1" Click="buttonAceptar_Click" IsEnabled="True" IsDefault="True" />
        <Button Content="Cancelar" Height="28" Width="75" HorizontalAlignment="Center"
Margin="0,0,0,0" Name="buttonCancelar" VerticalAlignment="Center" Grid.Column="1"
Grid.Row="1" Click="buttonCancelar_Click"  IsEnabled="True" IsCancel="True" />
    </Grid>
</Window>
```

13.4.5 PleoProg.Interfaz.Controles

13.4.5.1 ControlEvento.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using PleoProg.Estructuras;
using PleoProg.Datos;
using PleoProg.GestorPersonalidad;

namespace PleoProg.Interfaz.Controles
{
    /// <summary>
    /// Clase del Control ControlEvento
    /// </summary>
    public partial class ControlEvento : UserControl
    {
        /// <summary>
        /// Constructor del ControlEvento
        /// </summary>
        /// <param name="secuencia">Secuencia a la que pertenece el evento que
representa</param>
        /// <param name="gestor">GestorDSL que lo gestiona</param>
        /// <param name="posControl">Posición donde se dibujará</param>
        public ControlEvento(Secuencia secuencia, GestorDSL gestor,Point posControl)
        {
            InitializeComponent();
            lineaSig = null;
            this.gestor = gestor;
            this.secuencia = secuencia;
            this.evento = secuencia.evento;
            this.posControl = posControl;
            inicializarDatos();
        }

        /// <summary>
        /// Secuencia  a la que pertenece el control
        /// </summary>
        public Secuencia secuencia
        {
            get;
            private set;
        }

        /// <summary>
        /// Evento que contiene el control
        /// </summary>
        public Evento evento
        {
            get;
            private set;
        }

        /// <summary>
        /// Linea a la primera orden
        /// </summary>
        public Line lineaSig
        {
            get;
            set;
        }

        /// <summary>
```

```
        /// Posición del control
        /// </summary>
        public Point posControl
        {
            get;
            set;
        }

        /// <summary>
        /// Gestor
        /// </summary>
        private GestorDSL gestor
        {
            get;
            set;
        }

        /// <summary>
        /// Indica si el control está seleccionado
        /// </summary>
        private bool bSleccionado
        {
            get;
            set;
        }

        /// <summary>
        /// Se modifica el aspecto del control
        /// </summary>
        private void inicializarDatos()
        {
            textoEvento.Text = "";
            //Según el evento que sea se muestra un texto
            textoEvento.Text = Util.getNombreEvento(evento.iId);
        }

        /// <summary>
        /// Se pinta de rojo el evento seleccionado
        /// </summary>
        public void seleccionar()
        {
            this.bSleccionado = true;
            this.ellipseEvento.Stroke = Brushes.Red;
            this.labelEvento.Foreground = Brushes.Red;
        }

        /// <summary>
        /// Se pinta de negro el evento deseleccionado
        /// </summary>
        public void deseleccionar()
        {
            this.bSleccionado = false;
            this.ellipseEvento.Stroke = Brushes.Black;
            this.labelEvento.Foreground = Brushes.Black;
        }

        /// <summary>
        /// Al pulsar sobre el control con el ratón se colorea para hacer saber que está
seleccionado y
        /// se comienza a capturar la posición del ratón para que en los siguientes
eventos de movimiento de ratón se
        /// cambie también la posición del control
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void userControl_MouseDown(object sender, MouseButtonEventArgs e)
        {
            this.CaptureMouse();
            if (!bSleccionado)
            {
                this.seleccionar();
                if (gestor.eventoSeleccionado != null)
                    gestor.eventoSeleccionado.deseleccionar();

                if (gestor.ordenSeleccionada != null)
                    gestor.ordenSeleccionada.deseleccionar();

                gestor.eventoSeleccionado = this;
```

```
                gestor.eventoActual = this;
                gestor.ordenSeleccionada = null;
                gestor.seleccionarElementoComboBox(evento.iId);
            }
            posControl = e.GetPosition(gestor.canvas);
        }

        /// <summary>
        /// Al levantar el raton se deja de capturar el movimiento.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void userControl_MouseUp(object sender, MouseButtonEventArgs e)
        {
            this.ReleaseMouseCapture();
            gestor.canvas.InvalidateMeasure();
            //Se captura el evento, ya ha sido tratado
            e.Handled = true;
        }

        /// <summary>
        /// Al mover el ratón, si esta seleccionado el control, se mueve este también
teniendo en cuenta los límites
        /// del entorno de trabajo
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void userControl_MouseMove(object sender, MouseEventArgs e)
        {
            Point posCanvas = e.GetPosition(gestor.canvas);
            if (e.LeftButton == MouseButtonState.Pressed)
            {
                Double posXControl = Canvas.GetLeft(this) + (posCanvas.X -
posControl.X);
                Double posYControl = Canvas.GetTop(this) + (posCanvas.Y - posControl.Y);

                //No permite colocar fuera de los límites
                posXControl = posXControl < 0 ? 0 : posXControl;
                posYControl = posYControl < 0 ? 0 : posYControl;

                posXControl = posXControl > gestor.canvas.Width ? gestor.canvas.Width :
posXControl;
                posYControl = posYControl > gestor.canvas.Height ? gestor.canvas.Height
: posYControl;

                Canvas.SetLeft(this, posXControl);
                Canvas.SetTop(this, posYControl);

                gestor.reDibujarLinea(this, new Point(posXControl, posYControl));
            }
            posControl = posCanvas;
        }

        /// <summary>
        /// Al hacer doble clic se modifica el evento que se guarda
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void userControl_MouseDoubleClick(object sender, MouseButtonEventArgs e)
        {
            int iAntiguoId = evento.iId;

            WindowEvento WE = new WindowEvento(evento,gestor);
            if (WE.ShowDialog()==true)
                inicializarDatos();

            gestor.eliminarElementoComboBox(iAntiguoId);
            gestor.añadirElementoComboBox(evento.iId);
            gestor.seleccionarElementoComboBox(evento.iId);

            e.Handled = true;
        }

    }
}
```

13.4.5.2 ControlEvento.xaml

```
<UserControl x:Class="PleoProg.Interfaz.Controles.ControlEvento"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="130" d:DesignWidth="130" MouseDown="userControl_MouseDown"
MouseMove="userControl_MouseMove" MouseDoubleClick="userControl_MouseDoubleClick"
MouseUp="userControl_MouseUp">
    <Grid>
        <Ellipse Height="130" Width="130" HorizontalAlignment="Center" Margin="0"
Name="ellipseEvento" Stroke="Black" VerticalAlignment="Center" StrokeThickness="5"
Fill="White" />
        <Label Height="55" HorizontalAlignment="Center" Margin="0,0,0,0"
Name="labelEvento" VerticalAlignment="Center" Width="120"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center" FontStyle="Normal"
FontWeight="Black" Background="{x:Null}" Foreground="Black" FontSize="10">
            <TextBlock Name="textoEvento" TextWrapping="Wrap" FontSize="12"
TextAlignment="Center" />
        </Label>
    </Grid>
</UserControl>
```

13.4.5.3 ControlOrden.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using PleoProg.Estructuras;
using PleoProg.Datos;
using PleoProg.Interfaz;
using PleoProg.Interfaz.Controles;
using System.IO;
using PleoProg.GestorPersonalidad;

namespace PleoProg.Interfaz.Controles
{
    /// <summary>
    /// Clase del control ControlOrden
    /// </summary>
    public partial class ControlOrden : UserControl
    {
        /// <summary>
        /// Constructor de la clase ControlOrden
        /// </summary>
        /// <param name="secuencia">Secuencia a la que pertenece la Orden que
representa</param>
        /// <param name="orden">Orden que representa</param>
        /// <param name="gestor">GestorDSL que lo gestiona</param>
        /// <param name="posControl">Posición donde se dibujará</param>
        public ControlOrden(Secuencia secuencia,Orden orden, GestorDSL gestor,Point
posControl)
        {
            InitializeComponent();

            this.orden = orden;
            this.gestor = gestor;
            this.secuencia = secuencia;
            this.posControl = posControl;
            inicializarDatos();
        }
```

```
        /// <summary>
        /// Orden que representa el control
        /// </summary>
        public Orden orden
        {
            get;
            private set;
        }

        /// <summary>
        /// Secuencia de la orden que representa el control
        /// </summary>
        public Secuencia secuencia
        {
            get;
            private set;
        }

        /// <summary>
        /// Linea a la siguiente orden
        /// </summary>
        public Line lineaSig
        {
            get;
            set;
        }

        /// <summary>
        /// Linea a la anterior orden o evento
        /// </summary>
        public Line lineaAnt
        {
            get;
            set;
        }

        /// <summary>
        /// Posición del control
        /// </summary>
        public Point posControl
        {
            get;
            set;
        }

        /// <summary>
        /// Gestor de DSL
        /// </summary>
        private GestorDSL gestor
        {
            get;
            set;
        }

        /// <summary>
        /// Indica si el control está seleccionado
        /// </summary>
        private bool bSeleccionado
        {
            get;
            set;
        }

        /// <summary>
        /// Configura el aspecto inicial del control según la orden que represente
        /// </summary>
        private void inicializarDatos()
        {
            if (orden.GetType() == typeof(Movimiento))
            {
                Movimiento m = (Movimiento)orden;
                if(m.bRepetitivo)
                {
                    string datos="Grados: " + m.iGradosInicio + "\nGrados Fin: " +
m.iGradosFin + "\nRepeticiones: " + m.iRepeticiones;
                    mostrarObservaciones(datos);
                    labelTipoOrden.Content = "Repetición";
                    labelObservaciones.ToolTip = datos;
```

```
                }
                else
                {
                     string datos = "Grados: " + m.iGradosFin;
                mostrarObservaciones(datos);
                labelTipoOrden.Content = "Movimiento";
                labelObservaciones.ToolTip = datos;
                }

            }


            else if (orden.GetType() == typeof(CapturaImagen))
            {
                CapturaImagen cI = (CapturaImagen)orden;
                string datos = "Archivo: " + cI.sNombreArchivo;
                mostrarObservaciones(datos);
                this.rectangleObservaciones.Margin = new Thickness(0, 71, 0, 0);
                this.labelObservaciones.Margin = new Thickness(0, 71, 0, 0);
                labelTipoOrden.Content = "Foto";
                labelObservaciones.ToolTip = datos;
            }

            else if (orden.GetType() == typeof(CapturaSonido))
            {
                CapturaSonido cS = (CapturaSonido)orden;
                string datos = "Archivo: " + cS.sNombreArchivo+"\nTiempo:
"+cS.iSegundos;
                mostrarObservaciones(datos);
                this.rectangleObservaciones.Margin = new Thickness(0, 71, 0, 0);
                this.labelObservaciones.Margin = new Thickness(0, 71, 0, 0);
                labelTipoOrden.Content = "Grabar Sonido";
                labelObservaciones.ToolTip = datos;
            }

            else if (orden.GetType() == typeof(ReproducirSonido))
            {
                ReproducirSonido rS = (ReproducirSonido)orden;
                string datos = "Archivo: " +
System.IO.Path.GetFileName(rS.sRutaArchivo);
                mostrarObservaciones(datos);
                this.rectangleObservaciones.Margin = new Thickness(0, 71, 0, 0);
                this.labelObservaciones.Margin = new Thickness(0, 71, 0, 0);
                labelTipoOrden.Content = "Reproducir Sonido";
                labelObservaciones.ToolTip = rS.sRutaArchivo;
            }

            else if (orden.GetType() == typeof(Espera))
            {
                Espera espera = (Espera)orden;
                string datos = "Tiempo de espera: " + espera.iEspera;
                mostrarObservaciones(datos);
                this.rectangleObservaciones.Margin = new Thickness(0, 71, 0, 0);
                this.labelObservaciones.Margin = new Thickness(0, 71, 0, 0);
                labelTipoOrden.Content = "Tiempo de Espera";
                labelObservaciones.ToolTip = datos;
            }

            else if (orden.GetType() == typeof(Neutro))
            {
                ocultarObservaciones();
                labelTipoOrden.Content = "Pos. Normal";
            }

            else if (orden.GetType() == typeof(Motion))
            {
                Motion m = (Motion)orden;
                string datos = "Archivo: " +
System.IO.Path.GetFileName(m.sRutaAnimacion);
                mostrarObservaciones(datos);
                this.rectangleObservaciones.Margin = new Thickness(0, 71, 0, 0);
                this.labelObservaciones.Margin = new Thickness(0, 71, 0, 0);
                labelTipoOrden.Content = "Animación";
                labelObservaciones.ToolTip = m.sRutaAnimacion;
            }

            if (orden.GetType() == typeof(Movimiento))
            {
```

```
                mostrarMiembro(((Movimiento)orden).iMiembro);
            }
            else
            {
                ocultarMiembro();
            }
            labelTurno.Content = orden.iTurno;
            bSeleccionado = false;
        }

        /// <summary>
        /// Muestra Observaciones de la orden
        /// </summary>
        /// <param name="sDetalles">Cadena con los detalles</param>
        private void mostrarObservaciones(string sDetalles)
        {
            this.rectangleObservaciones.IsEnabled = true;
            this.rectangleObservaciones.Stroke = Brushes.Black;
            this.rectangleObservaciones.StrokeThickness = 3;
            this.rectangleObservaciones.Height = 65;
            this.rectangleObservaciones.Width = 150;

            this.labelObservaciones.IsEnabled = true;
            this.labelObservaciones.Background = Brushes.White;
            this.labelObservaciones.Height = 65;
            this.labelObservaciones.Width = 150;
            this.labelObservaciones.Content = sDetalles;
        }

        /// <summary>
        /// Muestra el nombre del miembro implicado
        /// </summary>
        /// <param name="iMiembro"></param>
        private void mostrarMiembro(int iMiembro)
        {
            labelMiembro.Content=Util.getNombreMiembro(iMiembro);

            this.labelMiembro.IsEnabled = true;
            this.labelMiembro.Background = Brushes.White;
            this.labelMiembro.Height = 30;
            this.labelMiembro.Width = 150;

            this.rectangleMiembro.IsEnabled = true;
            this.rectangleMiembro.Stroke = Brushes.Black;
            this.rectangleMiembro.StrokeThickness = 3;
            this.rectangleMiembro.Height = 32;
            this.rectangleMiembro.Width = 150;
        }

        /// <summary>
        /// Oculta las observaciones si la orden representada no las requiere
        /// </summary>
        private void ocultarObservaciones()
        {
            this.rectangleObservaciones.IsEnabled = false;
            this.rectangleObservaciones.Stroke = null;
            this.rectangleObservaciones.Height = 0;
            this.rectangleObservaciones.Width = 0;

            this.labelObservaciones.IsEnabled = false;
            this.labelObservaciones.Background = Brushes.Transparent;
            this.labelObservaciones.Height = 0;
            this.labelObservaciones.Width = 0;
            this.labelObservaciones.Content = "";
        }

        /// <summary>
        /// Oculta el espacio reservado para mostrar el nombre del miembro si no hay
ninguno implicado
        /// </summary>
        private void ocultarMiembro()
        {
            this.labelMiembro.IsEnabled = false;
            this.labelMiembro.Background = Brushes.Transparent;
            this.labelMiembro.Height = 0;
            this.labelMiembro.Width = 0;
            this.labelMiembro.Content = "";
```

```
            this.rectangleMiembro.IsEnabled = false;
            this.rectangleMiembro.Stroke = null;
            this.rectangleMiembro.Height = 0;
            this.rectangleMiembro.Width = 0;
        }

        /// <summary>
        /// Al pusar sobre el control con el ratón se comienza a capturar los
movimientos mientras no se deje de pulsar
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void userControl_MouseDown(object sender, MouseButtonEventArgs e)
        {
            this.CaptureMouse();
            if (!bSeleccionado)
            {
                //Al seleccionar una orden se busca el evento que la lanza para que sea
el activo
                Secuencia sec = gestor.personalidad.buscarSecuencia(orden);
                ControlEvento cE = gestor.buscarControlEvento(sec.evento);
                gestor.eventoActual = cE;
                gestor.seleccionarElementoComboBox(cE.evento.iId);

                this.seleccionar();
                if (gestor.ordenSeleccionada != null)
                    gestor.ordenSeleccionada.deseleccionar();
                if(gestor.eventoSeleccionado!=null)
                    gestor.eventoSeleccionado.deseleccionar();

                gestor.ordenSeleccionada = this;
                gestor.eventoSeleccionado = null;
                gestor.eventoActual = gestor.buscarControlEvento(this.secuencia.evento);
            }

            posControl = e.GetPosition(gestor.canvas);
        }

        /// <summary>
        /// Al mover el ratón si los movimientos están siendo capturados se mueve el
control
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void userControl_MouseMove(object sender, MouseEventArgs e)
        {
            Point posCanvas = e.GetPosition(gestor.canvas);
            if (e.LeftButton == MouseButtonState.Pressed)
            {
                Double posXControl = Canvas.GetLeft(this) + (posCanvas.X -
posControl.X);
                Double posYControl = Canvas.GetTop(this) + (posCanvas.Y - posControl.Y);

                //No permite colocar fuera de los límites
                posXControl = posXControl < 0 ? 0 : posXControl;
                posYControl = posYControl < 0 ? 0 : posYControl;

                posXControl = posXControl > gestor.canvas.Width ? gestor.canvas.Width :
posXControl;
                posYControl = posYControl > gestor.canvas.Height ? gestor.canvas.Height
: posYControl;

                Canvas.SetLeft(this, posXControl);
                Canvas.SetTop(this, posYControl);

                gestor.reDibujarLinea(this, new Point(posXControl, posYControl));
            }
            posControl = posCanvas;
        }

        /// <summary>
        /// Al solar el control se deja de capturar los movimientos
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void userControl_MouseUp(object sender, MouseButtonEventArgs e)
        {
            this.ReleaseMouseCapture();
```

```
            gestor.canvas.InvalidateMeasure();
            e.Handled = true;
        }

        /// <summary>
        /// Al hacer doble clic sobre el control se edita
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void userControl_MouseDoubleClick(object sender, MouseButtonEventArgs e)
        {
            this.ReleaseMouseCapture();
            //lanzar ventana apropiada, cargar valores actuales y recoger valores usados
            if (orden.GetType() == typeof(Movimiento))
            {
                WindowMovimiento WM = new WindowMovimiento((Movimiento)orden);
                if (WM.ShowDialog() == true)
                {
                    this.orden = WM.orden;
                    inicializarDatos();
                }
            }


            else if (orden.GetType() == typeof(CapturaImagen))
            {
                WindowNombreArchivoOrden WNAImagen = new
WindowNombreArchivoOrden((CapturaImagen)orden);
                if (WNAImagen.ShowDialog() == true)
                {
                    inicializarDatos();
                }
            }

            else if (orden.GetType() == typeof(CapturaSonido))
            {
                WindowNombreArchivoOrden WNASonido = new
WindowNombreArchivoOrden((CapturaSonido)orden);
                if (WNASonido.ShowDialog() == true)
                {
                    inicializarDatos();
                }
            }

            else if (orden.GetType() == typeof(ReproducirSonido))
            {
                ReproducirSonido rS = ((ReproducirSonido)this.orden);
                System.Windows.Forms.OpenFileDialog oFD = new
System.Windows.Forms.OpenFileDialog();

                oFD.InitialDirectory = rS.sRutaArchivo;
                oFD.Filter = "Waveform Audio (*.wav) | *.wav";
                oFD.FilterIndex = 0;
                oFD.RestoreDirectory = true;
                oFD.ShowDialog();

                if (oFD.FileName != "")
                {
                    string sNombre = oFD.FileName;
                    rS.sRutaArchivo = sNombre;
                    inicializarDatos();
                }
            }

            else if (orden.GetType() == typeof(Espera))
            {
                WindowTiempoEspera WEspera = new WindowTiempoEspera((Espera)orden);
                if (WEspera.ShowDialog() == true)
                {
                    inicializarDatos();
                }
            }
            e.Handled = true;
        }

        /// <summary>
        /// Al seleccionado el control se subraya en rojo
        /// </summary>
```

```
        public void seleccionar()
        {
            this.bSeleccionado = true;
            this.ellipseTurno.Stroke = Brushes.Red;
            this.labelTurno.Foreground = Brushes.Red;
            this.rectangleOrden.Stroke = Brushes.Red;
            this.labelTipoOrden.Foreground = Brushes.Red;
            this.rectangleMiembro.Stroke = Brushes.Red;
            this.labelMiembro.Foreground = Brushes.Red;
            this.rectangleObservaciones.Stroke = Brushes.Red;
            this.labelObservaciones.Foreground = Brushes.Red;
        }

        /// <summary>
        /// Al deseleccionar el control vuelve a su estado original
        /// </summary>
        public void deseleccionar()
        {
            this.bSeleccionado = false;
            this.ellipseTurno.Stroke = Brushes.Black;
            this.labelTurno.Foreground = Brushes.Black;
            this.rectangleOrden.Stroke = Brushes.Black;
            this.labelTipoOrden.Foreground = Brushes.Black;
            this.rectangleMiembro.Stroke = Brushes.Black;
            this.labelMiembro.Foreground = Brushes.Black;
            this.rectangleObservaciones.Stroke = Brushes.Black;
            this.labelObservaciones.Foreground = Brushes.Black;
        }
    }
}
```

13.4.5.4 ControlOrden.xaml

```
<UserControl x:Class="PleoProg.Interfaz.Controles.ControlOrden"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="173" d:DesignWidth="173" FontWeight="Normal"
MouseDown="userControl_MouseDown" MouseMove="userControl_MouseMove"
MouseDoubleClick="userControl_MouseDoubleClick" MouseUp="userControl_MouseUp">

    <Grid Height="173" Width="173">
        <Rectangle Height="50" HorizontalAlignment="Center" Margin="0,23,0,0"
Name="rectangleOrden" Stroke="Black" VerticalAlignment="Top" Width="150"
StrokeThickness="3" Fill="White" />
        <Rectangle Height="32" HorizontalAlignment="Center" Margin="0,71,0,0"
Name="rectangleMiembro" Stroke="Black" VerticalAlignment="Top" Width="150"
StrokeThickness="3" Fill="White" />
        <Rectangle Height="65" HorizontalAlignment="Center" Margin="0,98,0,0"
Name="rectangleObservaciones" Stroke="Black" VerticalAlignment="Top" Width="150"
StrokeThickness="3" Fill="White" />
        <Ellipse Height="42" HorizontalAlignment="Center" Margin="0" Name="ellipseTurno"
Stroke="Black" VerticalAlignment="Top" Width="42" StrokeThickness="5" Fill="White" />

        <Label Height="42" HorizontalAlignment="Center" Margin="0,0,0,0"
Name="labelTurno" VerticalAlignment="Top" Width="42" VerticalContentAlignment="Center"
HorizontalContentAlignment="Center" FontStyle="Normal" FontWeight="Black"
Background="{x:Null}" Foreground="Black" />
        <Label Height="27" HorizontalAlignment="Center" Margin="0,38,0,0"
Name="labelTipoOrden" VerticalAlignment="Top" Width="130"
HorizontalContentAlignment="Center" VerticalContentAlignment="Bottom"
FontWeight="Medium" FontSize="13" Background="{x:Null}" />
        <Label Height="65" HorizontalAlignment="Center" Margin="0,98,0,0"
Name="labelObservaciones" VerticalAlignment="Top" Width="150" BorderThickness="3"
Background="{x:Null}" VerticalContentAlignment="Center" />
        <Label Height="30" HorizontalAlignment="Center" Margin="0,71,0,0"
Name="labelMiembro" VerticalAlignment="Top" Width="150" BorderThickness="3"
VerticalContentAlignment="Center" HorizontalContentAlignment="Center"
FontWeight="Medium" Background="{x:Null}" />
    </Grid>
</UserControl>
```

13.4.6 PleoProg.Estructuras

13.4.6.1 Evento.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace PleoProg.Estructuras
{
    /// <summary>
    /// Clase que almacena un evento
    /// </summary>
    [Serializable]
    public class Evento
    {
        /// <summary>
        /// Constructor de la clase EVento.
        /// </summary>
        /// <param name="iId">ID del evento</param>
        public Evento(int iId)
        {
            this.iId = iId;
        }

        /// <summary>
        /// Constructor de la clase Evento. ID por defecto -1
        /// </summary>
        public Evento() { iId = -1; }

        /// <summary>
        /// Id. del evento, indica el tipo de evento
        /// </summary>
        public int iId
        {
            get;
            set;
        }

        /// <summary>
        /// Compara dos Eventos para determinar si son iguales
        /// </summary>
        /// <param name="evento">Evento con el que comparar</param>
        /// <returns>true, si son iguales</returns>
        public override bool Equals(Object evento)
        {
            Evento ev = (Evento)evento;
            if (ev.iId == this.iId)
                return true;
            return false;
        }

    }
}
```

13.4.6.2 Util.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace PleoProg.Datos
{
    static class Util
    {
        // Órdenes
        //      Movimientos
        public const int movHombroDerecho = 0;
        public const int movCodoDerecho = 1;
        public const int movHombroIzquierdo = 2;
        public const int movCodoIzquierdo = 3;
        public const int movCaderaIzquierda = 4;
        public const int movRodillaIzquierda = 5;
        public const int movCaderaDerecha = 6;
        public const int movRodillaDerecha = 7;
        public const int movTorso = 8;
        public const int movColaHorizontal = 9;
        public const int movColaVertical = 10;
        public const int movCuelloHorizontal = 11;
        public const int movCuelloVertical = 12;
        public const int movOjos = 13;
        public const int posNeutra = 99;
        // Otras ordenes
        public const int capturaFoto = 101;
        public const int capturaSonido = 102;
        public const int repSonido = 103;
        public const int tiempoEspera = 104;

        // Eventos
        // Inicio
        public const int eventoStart = 0;
        // Boton Wake Up
        public const int eventoWakeUp = 1;

        // Tactciles
        public const int eventoTactilCabeza = 2;
        public const int eventoTactilMenton= 3;
        public const int eventoTactilEspalda = 4;
        public const int eventoTactilCuello = 5;

        public const int eventoTactilExtremidadDelanteraDerecha = 6;
        public const int eventoTactilExtremidadDelanteraIzquierda = 7;
        public const int eventoTactilExtremidadTraseraDerecha = 8;
        public const int eventoTactilExtremidadTraseraIzquierda = 9;

        // Interruptores
        public const int eventoInterruptorExtremidadDelanteraDerecha = 10;
        public const int eventoInterruptorExtremidadDelanteraIzquierda = 11;
        public const int eventoInterruptorExtremidadTraseraDerecha = 12;
        public const int eventoInterruptorExtremidadTraseraIzquierda = 13;

        // Otros eventos
        public const int eventoObstaculo = 14;
        public const int eventoFilo = 15;
        public const int eventoSonido = 16;
        public const int eventoPosado = 17;
        public const int eventoAgitado = 18;
        public const int eventoAbuso = 19;
        public const int eventoAlzado = 20;
        public const int eventoObjetoBoca = 21;
        public const int eventoVariacionLuz = 22;
        public const int eventoOtroPleo = 23;

        // Constantes medidas Control Evento
        public const int anchoControlEvento = 130;
        public const int altoControlEvento = 130;
        // Constantes medidas Control Orden
        public const int anchoControlOrden = 175;
        public const int altoControlOrden = 175;
```

```
        /// <summary>
        /// Devuelve una cadena con el evento que coincide con el código que se la pasa
        /// </summary>
        /// <param name="iIdEvento"></param>
        /// <returns></returns>
        public static string getNombreEvento(int iIdEvento)
        {
            string newEvento = null;
            switch (iIdEvento)
            {
                case Util.eventoStart:
                    newEvento = "ENCENDIDO";
                    break;
                case Util.eventoWakeUp:
                    newEvento = "BOTÓN WAKE UP";
                    break;
                case Util.eventoTactilCabeza:
                    newEvento = "ACARICIAR CABEZA";
                    break;
                case Util.eventoTactilMenton:
                    newEvento = "ACARICIAR MENTÓN";
                    break;
                case Util.eventoTactilEspalda:
                    newEvento = "ACARICIAR ESPALDA";
                    break;
                case Util.eventoTactilCuello:
                    newEvento = "ACARICIAR CUELLO";
                    break;
                case Util.eventoTactilExtremidadDelanteraDerecha:
                    newEvento = "ACARICIAR PATA DELANTERA DERECHA";
                    break;
                case Util.eventoTactilExtremidadDelanteraIzquierda:
                    newEvento = "ACARICIAR PATA DELANTERA IZQUIERDA";
                    break;
                case Util.eventoTactilExtremidadTraseraDerecha:
                    newEvento = "ACARICIAR PATA TRASERA DERECHA";
                    break;
                case Util.eventoTactilExtremidadTraseraIzquierda:
                    newEvento = "ACARICIAR PATA TRASERA IZQUIERDA";
                    break;
                case Util.eventoInterruptorExtremidadDelanteraDerecha:
                    newEvento = "INTERRUPTOR PATA DELANTERA DERECHA";
                    break;
                case Util.eventoInterruptorExtremidadDelanteraIzquierda:
                    newEvento = "INTERRUPTOR PATA DELANTERA IZQUIERDA";
                    break;
                case Util.eventoInterruptorExtremidadTraseraDerecha:
                    newEvento = "INTERRUPTOR PATA TRASERA DERECHA";
                    break;
                case Util.eventoInterruptorExtremidadTraseraIzquierda:
                    newEvento = "INTERRUPTOR PATA TRASERA IZQUIERDA";
                    break;
                case Util.eventoObstaculo:
                    newEvento = "OBSTÁCULO";
                    break;
                case Util.eventoFilo:
                    newEvento = "FILO";
                    break;
                case Util.eventoSonido:
                    newEvento = "SONIDO";
                    break;
                case Util.eventoPosado:
                    newEvento = "POSADO";
                    break;
                case Util.eventoAgitado:
                    newEvento = "AGITADO";
                    break;
                case Util.eventoAbuso:
                    newEvento = "ABUSO";
                    break;
                case Util.eventoAlzado:
                    newEvento = "ALZADO";
                    break;
                case Util.eventoObjetoBoca:
                    newEvento = "OBJETO EN LA BOCA";
                    break;
                case Util.eventoVariacionLuz:
```

```
                    newEvento = "VARIACIÓN DE LUZ";
                    break;
                case Util.eventoOtroPleo:
                    newEvento = "OTRO PLEO";
                    break;
            }
            return newEvento;
        }

        /// <summary>
        /// Devuelve una cadena con el nombre del evento utilizado por el compilador de
Pleo
        /// </summary>
        /// <param name="iIdEvento"></param>
        /// <returns></returns>
        public static string getNombreEventoCompiladorPleo(int iIdEvento)
        {
            string newEvento = null;
            switch (iIdEvento)
            {
                case Util.eventoWakeUp:
                    newEvento = "SENSOR_WAKEUP";
                    break;
                case Util.eventoTactilCabeza:
                    newEvento = "SENSOR_HEAD";
                    break;
                case Util.eventoTactilMenton:
                    newEvento = "SENSOR_CHIN";
                    break;
                case Util.eventoTactilEspalda:
                    newEvento = "SENSOR_TAIL";
                    break;
                case Util.eventoTactilCuello:
                    newEvento = "SENSOR_BACK";
                    break;
                case Util.eventoTactilExtremidadDelanteraDerecha:
                    newEvento = "SENSOR_RIGHT_ARM";
                    break;
                case Util.eventoTactilExtremidadDelanteraIzquierda:
                    newEvento = "SENSOR_LEFT_ARM";
                    break;
                case Util.eventoTactilExtremidadTraseraDerecha:
                    newEvento = "SENSOR_RIGHT_LEG";
                    break;
                case Util.eventoTactilExtremidadTraseraIzquierda:
                    newEvento = "SENSOR_LEFT_LEG";
                    break;
                case Util.eventoInterruptorExtremidadDelanteraDerecha:
                    newEvento = "SENSOR_FRONT_RIGHT";
                    break;
                case Util.eventoInterruptorExtremidadDelanteraIzquierda:
                    newEvento = "SENSOR_FRONT_LEFT";
                    break;
                case Util.eventoInterruptorExtremidadTraseraDerecha:
                    newEvento = "SENSOR_BACK_RIGHT";
                    break;
                case Util.eventoInterruptorExtremidadTraseraIzquierda:
                    newEvento = "SENSOR_BACK_LEFT";
                    break;
                case Util.eventoObstaculo:
                    newEvento = "SENSOR_OBJECT_IN_FRONT";
                    break;
                case Util.eventoFilo:
                    newEvento = "SENSOR_EDGE_IN_FRONT";
                    break;
                case Util.eventoSonido:
                    newEvento = "SENSOR_SOUND_LOUD";
                    break;
                case Util.eventoPosado:
                    newEvento = "SENSOR_PICKED_DOWN";
                    break;
                case Util.eventoAgitado:
                    newEvento = "SENSOR_SHAKE";
                    break;
                case Util.eventoAbuso:
                    newEvento = "SENSOR_ABUSE";
                    break;
                case Util.eventoAlzado:
```

```
                newEvento = "SENSOR_PICKED_UP";
                break;
            case Util.eventoObjetoBoca:
                newEvento = "SENSOR_MOUTH";
                break;
            case Util.eventoVariacionLuz:
                newEvento = "SENSOR_LIGHT_CHANGE";
                break;
            case Util.eventoOtroPleo:
                newEvento = "SENSOR_BEACON";
                break;
        }
        return newEvento;
    }

    /// <summary>
    /// Retorna el nombre del miembro identificado por el id que se le pasa
    /// </summary>
    /// <param name="iMiembro"></param>
    /// <returns></returns>
    public static string getNombreMiembro(int iMiembro)
    {
        string sMiembro = "";
        switch (iMiembro)
        {
            case (Util.movHombroDerecho):
                sMiembro = "HOMBRO DERECHO";
                break;
            case (Util.movHombroIzquierdo):
                sMiembro = "HOMBRO IZQUIERDO";
                break;

            case (Util.movRodillaDerecha):
                sMiembro = "RODILLA DERECHA";
                break;
            case (Util.movRodillaIzquierda):
                sMiembro = "RODILLA IZQUIERDA";
                break;

            case (Util.movCodoDerecho):
                sMiembro = "CODO DERECHO";
                break;
            case (Util.movCodoIzquierdo):
                sMiembro = "CODO IZQUIERDO";
                break;

            case (Util.movCaderaDerecha):
                sMiembro = "CADERA DERECHA";
                break;
            case (Util.movCaderaIzquierda):
                sMiembro = "CADERA IZQUIERDA";
                break;

            case (Util.movTorso):
                sMiembro = "TORSO";
                break;

            case (Util.movColaHorizontal):
                sMiembro = "COLA HORIZ.";
                break;
            case (Util.movColaVertical):
                sMiembro = "COLA VERT.";
                break;

            case (Util.movCuelloHorizontal):
                sMiembro = "CUELLO HORIZ.";
                break;
            case (Util.movCuelloVertical):
                sMiembro = "CUELLO VERT.";
                break;

            case (Util.movOjos):
                sMiembro = "OJOS";
                break;
        }
        return sMiembro;
    }
```

```
        /// <summary>
        /// Devuelve un código que se corresponde con la cadena que se inserte
        /// </summary>
        /// <param name="sEvento"></param>
        /// <returns></returns>
        public static int getCodigoEvento(string sEvento)
        {
            if (sEvento.CompareTo("ENCENDIDO") == 0)
                return Util.eventoStart;

            if (sEvento.CompareTo("BOTÓN WAKE UP") == 0)
                return Util.eventoWakeUp;

            if (sEvento.CompareTo("ACARICIAR CABEZA") == 0)
                return Util.eventoTactilCabeza;

            if (sEvento.CompareTo("ACARICIAR MENTÓN") == 0)
                return Util.eventoTactilMenton;

            if (sEvento.CompareTo("ACARICIAR ESPALDA") == 0)
                return Util.eventoTactilEspalda;

            if (sEvento.CompareTo("ACARICIAR CUELLO") == 0)
                return Util.eventoTactilCuello;

            if (sEvento.CompareTo("ACARICIAR PATA DELANTERA DERECHA") == 0)
                return Util.eventoTactilExtremidadDelanteraDerecha;

            if (sEvento.CompareTo("ACARICIAR PATA DELANTERA IZQUIERDA") == 0)
                return Util.eventoTactilExtremidadDelanteraIzquierda;

            if (sEvento.CompareTo("ACARICIAR PATA TRASERA DERECHA") == 0)
                return Util.eventoTactilExtremidadTraseraDerecha;

            if (sEvento.CompareTo("ACARICIAR PATA TRASERA IZQUIERDA") == 0)
                return Util.eventoTactilExtremidadTraseraIzquierda;

            if (sEvento.CompareTo("INTERRUPTOR PATA DELANTERA DERECHA") == 0)
                return Util.eventoInterruptorExtremidadDelanteraDerecha;

            if (sEvento.CompareTo("INTERRUPTOR PATA DELANTERA IZQUIERDA") == 0)
                return Util.eventoInterruptorExtremidadDelanteraIzquierda;

            if (sEvento.CompareTo("INTERRUPTOR PATA TRASERA DERECHA") == 0)
                return Util.eventoInterruptorExtremidadTraseraDerecha;

            if (sEvento.CompareTo("INTERRUPTOR PATA TRASERA IZQUIERDA") == 0)
                return Util.eventoInterruptorExtremidadTraseraIzquierda;

            if (sEvento.CompareTo("OBSTÁCULO") == 0)
                return Util.eventoObstaculo;

            if (sEvento.CompareTo("FILO") == 0)
                return Util.eventoFilo;

            if (sEvento.CompareTo("SONIDO") == 0)
                return Util.eventoSonido;

            if (sEvento.CompareTo("POSADO") == 0)
                return Util.eventoPosado;

            if (sEvento.CompareTo("AGITADO") == 0)
                return Util.eventoAgitado;

            if (sEvento.CompareTo("ABUSO") == 0)
                return Util.eventoAbuso;

            if (sEvento.CompareTo("ALZADO") == 0)
                return Util.eventoAlzado;

            if (sEvento.CompareTo("OBJETO EN LA BOCA") == 0)
                return Util.eventoObjetoBoca;

            if (sEvento.CompareTo("VARIACIÓN DE LUZ") == 0)
                return Util.eventoVariacionLuz;

            if (sEvento.CompareTo("OTRO PLEO") == 0)
                return Util.eventoOtroPleo;
```

```
            return -1;
        }
    }
}
```

13.4.7 PleoProg.Estructuras.Ordenes

13.4.7.1 Orden.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace PleoProg.Estructuras
{
    /// <summary>
    /// Clase Orden de la que heredan las demás ordenes
    /// </summary>
    [Serializable]
    public abstract class Orden
    {
        /// <summary>
        /// Constructor de la clase Orden
        /// </summary>
        /// <param name="iId">ID de la Orden</param>
        /// <param name="iTurno">Turno de la Orden</param>
        public Orden(int iId, int iTurno)
        {
            this.iId = iId;
            this.iTurno = iTurno;
        }

        /// <summary>
        /// Turno de la orden
        /// </summary>
        public int iTurno
        {
            get;
            set;
        }

        /// <summary>
        /// Id de la orden
        /// </summary>
        public int iId
        {
            get;
            protected set;
        }

        /// <summary>
        /// Comprueba si una Orden es igual a la actual
        /// </summary>
        /// <param name="o">Orden con la que comparar</param>
        /// <returns>true, si es la misma orden</returns>
        public override bool Equals(Object o)
        {
            Orden orden = (Orden)o;
            if (orden.iId == this.iId)
                return true;
            return false;
        }

        /// <summary>
        /// Escribe en el flujo que se le pase un trozo de código XML correspondiente a
la orden
        /// </summary>
        /// <param name="sW">StreamWriter donde se escribirá</param>
```

```
        public abstract void toXML(StreamWriter sW);
    }
}
```

13.4.7.2 CapturaImagen.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;


namespace PleoProg.Estructuras
{
    /// <summary>
    /// Clase de la orden CapturaImagen
    /// </summary>
    [Serializable]
    public class CapturaImagen : Orden
    {
        /// <summary>
        /// Constructor de la clase.
        /// </summary>
        /// <param name="iId">ID de la orden</param>
        /// <param name="iTurno">Turno de la orden</param>
        /// <param name="sNombreArchivo">Nombre del archivo de imagen a guardar</param>
        public CapturaImagen(int iId, int iTurno, string sNombreArchivo) : base(iId,
iTurno)
        {
            this.sNombreArchivo = sNombreArchivo;
        }

        /// <summary>
        /// Nombre del archivo que se le dará a la imagen tomada
        /// </summary>
        public string sNombreArchivo
        {
            get;
            set;
        }

        /// <summary>
        /// Escribe en el flujo que se le pase un trozo de código XML correspondiente a
la orden
        /// </summary>
        /// <param name="sW">StreamWriter donde se escribirá la información</param>
        public override void toXML(StreamWriter sW)
        {
            sW.WriteLine("\t\t\t<Orden>");
            sW.WriteLine("\t\t\t\t<Tipo>Captura Imagen</Tipo>");
            sW.WriteLine("\t\t\t\t<Id>" + this.iId + "</Id>");
            sW.WriteLine("\t\t\t\t<Turno>" + this.iTurno + "</Turno>");
            sW.WriteLine("\t\t\t\t<Nombre>" + this.sNombreArchivo + "</Nombre>");
            sW.WriteLine("\t\t\t</Orden>");
        }
    }
}
```

13.4.7.3 CapturaSonido.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace PleoProg.Estructuras
{
    /// <summary>
    /// Clase de la orden CapturaSonido
    /// </summary>
    [Serializable]
    public class CapturaSonido : Orden
    {
        /// <summary>
        /// Constructor de la clase CapturaSonido
        /// </summary>
        /// <param name="iId">ID de la orden</param>
        /// <param name="iTurno">Turno de la orden</param>
        /// <param name="sNombreArchivo">Nombre del archivo de sonido que se
guardará</param>
        /// <param name="iSegundos">Duaración de la grabación</param>
        public CapturaSonido(int iId, int iTurno, string sNombreArchivo, int iSegundos)
: base(iId, iTurno)
        {
            this.sNombreArchivo = sNombreArchivo;
            this.iSegundos = iSegundos;
        }

        /// <summary>
        /// Nombre del archivo que se almacenará
        /// </summary>
        public string sNombreArchivo
        {
            get;
            set;
        }

        /// <summary>
        /// Duración del archivo que se almacenará
        /// </summary>
        public int iSegundos
        {
            get;
            set;
        }

        /// <summary>
        /// Escribe en el flujo que se le pase un trozo de código XML correspondiente a
la orden
        /// </summary>
        /// <param name="sW">StreamWriter donde se escribirá la información</param>
        public override void toXML(StreamWriter sW)
        {
            sW.WriteLine("\t\t\t<Orden>");
            sW.WriteLine("\t\t\t\t<Tipo>Captura Sonido</Tipo>");
            sW.WriteLine("\t\t\t\t<Id>" + this.iId + "</Id>");
            sW.WriteLine("\t\t\t\t<Turno>" + this.iTurno + "</Turno>");
            sW.WriteLine("\t\t\t\t<Nombre>" + this.sNombreArchivo + "</Nombre>");
            sW.WriteLine("\t\t\t\t<Tiempo>" + this.iSegundos + "</Tiempo>");
            sW.WriteLine("\t\t\t</Orden>");
        }
    }
}
```

13.4.7.4 Espera.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace PleoProg.Estructuras
{
    /// <summary>
    /// Clase de la orden Espera
    /// </summary>
    [Serializable]
    public class Espera : Orden
    {
        /// <summary>
        /// Constructor de la clase Espera
        /// </summary>
        /// <param name="iId">ID de la Orden</param>
        /// <param name="iTurno">Turno de la Orden</param>
        /// <param name="iEspera">Tiempo, en segundos, de espera</param>
        public Espera(int iId, int iTurno, int iEspera)
            : base(iId, iTurno)
        {
            this.iEspera = iEspera;
        }

        /// <summary>
        /// Tiempo de espera
        /// </summary>
        public int iEspera
        {
            get;
            set;
        }

        /// <summary>
        /// Escribe en el flujo que se le pase un trozo de código XML correspondiente a
la orden
        /// </summary>
        /// <param name="sW">StreamWriter donde se escribirá la información</param>
        public override void toXML(StreamWriter sW)
        {
            sW.WriteLine("\t\t\t<Orden>");
            sW.WriteLine("\t\t\t\t<Tipo>Espera</Tipo>");
            sW.WriteLine("\t\t\t\t<Id>" + iId + "</Id>");
            sW.WriteLine("\t\t\t\t<Turno>" + iTurno + "</Turno>");
            sW.WriteLine("\t\t\t\t<Tiempo>" + iEspera + "</Tiempo>");
            sW.WriteLine("\t\t\t</Orden>");
        }
    }
}
```

13.4.7.5 Motion.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace PleoProg.Estructuras
{
    /// <summary>
    /// Clase de la orden Animación
    /// </summary>
    [Serializable]
    class Motion : Orden
    {
        public Motion(int iId, int iTurno, string sRutaAnimacion) : base(iId, iTurno)
        {
```

```
            this.sRutaAnimacion = sRutaAnimacion;
        }

        /// <summary>
        /// Ruta del archivo .csv que se ejecutará
        /// </summary>
        public string sRutaAnimacion
        {
            get;
            set;
        }

        /// <summary>
        /// Escribe en el flujo que se le pase un trozo de código XML correspondiente a
la orden
        /// </summary>
        /// <param name="sW">StreamWriter donde se escribirá la información</param>
        public override void toXML(StreamWriter sW)
        {
            sW.WriteLine("\t\t\t<Orden>");
            sW.WriteLine("\t\t\t\t<Tipo>Motion</Tipo>");
            sW.WriteLine("\t\t\t\t<Id>" + iId + "</Id>");
            sW.WriteLine("\t\t\t\t<Turno>" + iTurno + "</Turno>");
            sW.WriteLine("\t\t\t\t<Ruta>" + sRutaAnimacion + "</Ruta>");
            sW.WriteLine("\t\t\t</Orden>");
        }
    }
}
```

13.4.7.6 Movimiento.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace PleoProg.Estructuras
{
    /// <summary>
    /// Clase de la orden Movimiento
    /// </summary>
    [Serializable]
    public class Movimiento : Orden
    {
        /// <summary>
        /// Constructor de la clase Movimiento.
        /// </summary>
        /// <param name="uId">ID de la Orden</param>
        /// <param name="uTurno">Turno de la Orden</param>
        /// <param name="iGradosInicio">Grados de inicio del movimiento</param>
        /// <param name="iGradosFin">Grados de fin del movimiento</param>
        /// <param name="iRepeticiones">Número de repeticiones</param>
        /// <param name="iMiembro">Miembro a mover</param>
        /// <param name="bRepetitivo">true, si es un movimiento repetitivo</param>
        public Movimiento(int uId, int uTurno, int iGradosInicio, int iGradosFin, int
iRepeticiones, int iMiembro, bool bRepetitivo)
            : base(uId, uTurno)
        {
            this.iMiembro = iMiembro;

            this.iGradosInicio = iGradosInicio;
            this.iGradosFin = iGradosFin;

            this.bRepetitivo = bRepetitivo;
            this.iRepeticiones = iRepeticiones;
        }

        /// <summary>
        /// Entero que identifica el miembro a mover
        /// </summary>
        public int iMiembro
        {
            get;
```

```
            set;
        }

        /// <summary>
        /// Grados de inicio de movimiento
        /// </summary>
        public int iGradosInicio
        {
            get;
            set;
        }

        /// <summary>
        /// Grados de fin de movimiento
        /// </summary>
        public int iGradosFin
        {
            get;
            set;
        }

        /// <summary>
        /// Numero de repeticiones
        /// </summary>
        public int iRepeticiones
        {
            get;
            set;
        }

        /// <summary>
        /// Indica si se trata de un movimiento repetitivo
        /// </summary>
        public bool bRepetitivo
        {
            get;
            set;
        }

        /// <summary>
        /// Escribe en el flujo que se le pase un trozo de código XML correspondiente a
la orden
        /// </summary>
        /// <param name="sW">StreamWriter donde se escribirá la información</param>
        public override void toXML(StreamWriter sW)
        {
            sW.WriteLine("\t\t\t<Orden>");
            sW.WriteLine("\t\t\t\t<Id>" + iId + "</Id>");
            sW.WriteLine("\t\t\t\t<Turno>" + iTurno + "</Turno>");

            if (bRepetitivo == true)
                sW.WriteLine("\t\t\t\t<Tipo>Movimiento Repetitivo</Tipo>");
            else
                sW.WriteLine("\t\t\t\t<Tipo>Movimiento Único</Tipo>");


            sW.WriteLine("\t\t\t\t<Miembro>" + iMiembro + "</Miembro>");
            sW.WriteLine("\t\t\t\t<AnguloFin>" + iGradosFin + "</AnguloFin>");


            if (bRepetitivo == true)
            {
                sW.WriteLine("\t\t\t\t<AnguloInicio>" + iGradosInicio +
"</AnguloInicio>");
                sW.WriteLine("\t\t\t\t<Repeticiones>" + iRepeticiones +
"</Repeticiones>");
            }
            sW.WriteLine("\t\t\t</Orden>");
        }

    }
}
```

13.4.7.7 Neutro.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace PleoProg.Estructuras
{
    /// <summary>
    /// Clase de la orden posición neutra
    /// </summary>
    [Serializable]
    class Neutro : Orden
    {
        /// <summary>
        /// Constructor de la clase Neutro
        /// </summary>
        /// <param name="iId">ID de la orden</param>
        /// <param name="iTurno">Turno de la orden</param>
        public Neutro(int iId, int iTurno) : base(iId, iTurno) { }

        /// <summary>
        /// Escribe en el flujo que se le pase un trozo de código XML correspondiente a
la orden
        /// </summary>
        /// <param name="sW">StreamWriter donde se escribirá la información</param>
        public override void toXML(StreamWriter sW)
        {
            sW.WriteLine("\t\t\t<Orden>");
            sW.WriteLine("\t\t\t\t<Tipo>Neutro</Tipo>");
            sW.WriteLine("\t\t\t\t<Id>" + iId + "</Id>");
            sW.WriteLine("\t\t\t\t<Turno>" + iTurno + "</Turno>");
            sW.WriteLine("\t\t\t</Orden>");
        }
    }

}
```

13.4.7.8 ReproducirSonido.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace PleoProg.Estructuras
{
    /// <summary>
    /// Clase de la orden ReproducirSonido
    /// </summary>
    [Serializable]
    class ReproducirSonido : Orden
    {
        /// <summary>
        /// Constructor de la clase ReproducirSonido
        /// </summary>
        /// <param name="iId">ID de la orden</param>
        /// <param name="iTurno">Turno de la Orden</param>
        /// <param name="sRutaArchivo">Ruta del archivo que se reproducirá</param>
        public ReproducirSonido(int iId, int iTurno, string sRutaArchivo) : base(iId,
iTurno)
        {
            this.sRutaArchivo = sRutaArchivo;
        }

        /// <summary>
        /// Ruta del archivo que se reproducirá
        /// </summary>
        public string sRutaArchivo
        {
            get;
```

```
            set;
        }

        /// <summary>
        /// Escribe en el flujo que se le pase un trozo de código XML correspondiente a
la orden
        /// </summary>
        /// <param name="sW">StreamWriter donde se escribirá la información</param>
        public override void toXML(StreamWriter sW)
        {
            sW.WriteLine("\t\t\t<Orden>");
            sW.WriteLine("\t\t\t\t<Tipo>Reproducir Sonido</Tipo>");
            sW.WriteLine("\t\t\t\t<Id>" + iId + "</Id>");
            sW.WriteLine("\t\t\t\t<Turno>" + iTurno + "</Turno>");
            sW.WriteLine("\t\t\t\t<Ruta>" + sRutaArchivo + "</Ruta>");
            sW.WriteLine("\t\t\t</Orden>");
        }
    }
}
```