UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 8 y 9

TRABAJO DE DIPLOMA PARA OPTAR

POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

TÍTULO:

Análisis de un IDE para múltiples plataformas con tecnologías y herramientas libres para desarrollar software educativo en formato multimedia. Subsistema de gestión visual.

Autores:

Anisley de la Caridad Saez Villavicencio

Yonnys Pablo Martín Olivera

Tutor: Ing. Abel Ernesto Lorente Rodríguez

Ciudad de La Habana

Junio, 2008

AGRADECIMIENTOS

“Agradece a la llama su luz, pero no olvides el pie del candil que, constante y paciente, la sostiene en la sombra”

Rabindranath Tagore

De entre tres palabras mágicas “gracias”, ”perdón” y ”permiso”, haremos uso de la primera para intentar llegar a los corazones de todas aquellas personas que de una forma u otra han hecho posible que este momento se hiciera realidad.

A los padres que han sido la guía y el motor impulsor en la realización de nuestros sueños.

A los amigos que siempre han estado a nuestro lado, por el apoyo y la confianza brindada.

A la Revolución y al Comandante en Jefe por darnos la oportunidad de forjar un futuro mejor.

MUCHAS GRACIAS

DEDICATORIA

A mis padres, mi abuela y mis hermanos.

A todos los que siempre me han apoyado.

Yonnys

DECLARACIÓN DE AUTORÍA

Declaro ser el autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas el derecho de los valores patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los __ días del mes de _____ del año ____.

Anisley de la Caridad Saez Villavicencio

Yonnys Pablo Martín Olivera

_____

_____

Ing. Abel Ernesto Lorente Rodríguez

RESUMEN

En la actualidad las nuevas tecnologías de la información y las comunicaciones se han extendido a todos los rincones del mundo, y están causando un impacto revolucionario en los medios de enseñanza tradicionales. Con estos nuevos modelos educativos se va haciendo un énfasis mayor en las universidades y aulas virtuales, donde los estudiantes aprenden e interactúan con software educativos relacionados con las materias que reciben. En Cuba existe una creciente necesidad de productos educativos, entre ellos multimedia. Dichos productos se desarrollan muchas veces usando herramientas de autor propietarias sin licencias, lo que impide su comercialización y exportación. Es significativo destacar que la mayoría de estas herramientas solo permiten crear contenido para una sola plataforma (Microsoft Windows), esto rompe con la estrategia de migración a software libre que se ha trazado el país en vista de lograr la independencia tecnológica. Actualmente se encuentra en desarrollo un IDE multiplataforma bajo entornos libres para la creación de multimedia. Este trabajo de diploma presenta una herramienta que permitirá a los desarrolladores finales de multimedia diseñar interfaces visuales, garantizando todas las facilidades y funcionalidades de un editor visual moderno.

PALABRAS CLAVES: IDE, Herramientas Libres, Multimedia, Software Libre

ÍNDICE

INTRODUCCIÓN

El mundo avanza cada día a una velocidad mayor, las nuevas tecnologías quedan obsoletas en breves periodos de tiempo, soluciones que antes hubiesen llevado a la humanidad miles de años para su sustitución hoy apenas dan cabida para actualizarse. El mundo del software por su papel avanzado en el desarrollo tecnológico de la era moderna y el espectro educacional no se quedan atrás; cada uno vive cambios vertiginosos y se adapta a ellos. ¿Pero que papel juega el software en la esfera educacional? (ARECHABALETA, 2008) ¿Cómo este ha ido adquirido terreno para la formación del hombre del mañana?

A nivel mundial han surgido nuevos conceptos asociados a la Informática y la Educación. La creación de Aulas Virtuales (DÍAZ, 2007) para fomentar la educación a distancia y el desarrollo de plataformas de teleformación han generado un despunte acelerado de la función de ambas esferas. El país no se queda atrás en esta nueva visión social. En los albores de la Revolución la educación solo requería un lápiz, una libreta y un candil; sin embargo hoy los recursos técnicos de los centros educativos incluyen programas que sumergen a la población en la computación moderna desde edades tempranas.

La misión de informatizar al país es prioritaria y sobre esta base se creó la Universidad de las Ciencias Informáticas (UCI), para la cual la formación y la superación personal son guías de investigación y desarrollo. La universidad ha insertado el desarrollo de software educativo en la producción nacional de tal forma que los medios de enseñanza se han visto provistos de disimiles productos en los que apoyar el proceso de aprendizaje. Pero con las nuevas ideas surgen también viejos problemas que se trasladan a las nuevas situaciones.

El Embargo económico y político del Gobierno de los Estados Unidos de América hacia Cuba también ha afectado seriamente el desarrollo de soluciones informáticas para la educación. La mayor parte de las grandes compañías productoras de herramientas y materiales para este sector son de origen norteamericano; y las trabas que impone su gobierno para el uso libre de estos productos, lleva a la necesidad de incentivar la producción nacional de software educativo basándose en nuevas soluciones. Dichas soluciones deben resolver las necesidades de la nación, y porque no, también permitir la comercialización o instauración de productos cubanos en el exterior.

Pero otro tema surge a consideración, y es el de la libertad de software. Cuando alguien se refiere a libertad de software (STALLMAN, 2007) se habla de la libertad de uso, estudio, modificación y

distribución del mismo. Cuatro puntos marcados en su manifiesto de software libre por Richard Stallman como los esenciales principio éticos por lo que debe estar regido el desarrollo de software en la actualidad. En toda Cuba, se lleva a cabo un plan de migración a software libre y específicamente en la UCI este proceso de migración repercute negativamente en la producción de software educativo en formatos multimedia.

Para lograr establecer satisfactoriamente una migración gradual en este sentido, los desarrolladores en el área requieren de herramientas que ofrezcan soluciones completas con ambientes de desarrollo muy profesionales como el Adobe Flash (URLICH, 2002), o el proyecto de Microsoft: Expresión; herramientas que actualmente no existen bajo una licencia que permita usarlas libremente o no cubren las necesidades requeridas por los desarrolladores.

Debido a estos inconvenientes surge como un **problema a resolver** la inquietud de poder desarrollar por medios propios estas herramientas. Se requiere de una herramienta capaz de afrontar todos los retos que conlleva la creación de software educativo con los requisitos mínimos e indispensables. Herramienta que siga los principios antes propuestos sobre la libertad. Y que a su vez destaque e independice de todos los problemas que rodean estas cuestiones con su solución.

Una de las principales tareas a enfrentar será el estudio a fondo del espectro mundial que rodea a este problema en cuanto a: intentos anteriores y similares, posibilidades reales en el desarrollo del este tipo de software, y estudio de las alternativas que estén al alcance. Encaminando siempre este proceso como **objeto de estudio** al desarrollo de un editor visual para el soporte y el cumplimiento de las necesidades reales que existen hoy en día en la producción de software educativo con formatos multimedia de la Universidad. Teniendo como **campo de acción** en dicho problema el desarrollo de productos multimedia y software educativo en la UCI.

El **objetivo general** del trabajo es proporcionar una herramienta para la gestión visual de recursos multimedia en el Ambiente de Desarrollo Integrado (IDE) propuesto. Para darle cumplimiento al presente trabajo se cuenta con varios objetivos específicos tales como:

- Fundamentar la validez del proyecto a partir del análisis de otras soluciones existentes.
- Documentar las herramientas y tecnologías a utilizar en el editor visual de recursos multimedia.
- Definir una arquitectura que permita al subsistema adaptarse a distintos tipos de proyectos multimedia.
- Implementar una herramienta para la edición visual en el IDE propuesto.
- Difundir y categorizar el resultado del proyecto.

Para el desarrollo del mismo se cuenta con una serie de **tareas** que ayudarán a cumplimentar la construcción del mismo:

- Evaluar las diferentes soluciones encontradas a raíz de la investigación.
- Definir situación problemática, objetivos, aportes teóricos y prácticos e impacto social.
- Documentar las herramientas y tecnologías a utilizar en el editor visual de recursos multimedia.
- Efectuar un levantamiento de requisitos primarios para una propuesta del proyecto.
- Análisis de las posibles librerías libres para interfaz de usuario (GUI) que permiten la creación del subsistema.
- Selección y fundamentación de una metodología de desarrollo acorde a la magnitud del proyecto.
- Evaluación de los patrones de arquitectura y estándares de diseño a utilizar.
- Documentar, en base a la metodología de desarrollo seleccionada, el análisis del subsistema en cuestión según requisitos definidos anteriormente.
- Estudiar las técnicas de diseño gráfico para la construcción de la interfaz de usuario, así como la arquitectura de la información correspondiente.
- Implementación por etapas de un prototipo funcional, dependiendo de los subsistemas arrojados por el análisis del proyecto, que cubra los requisitos de mayor prioridad.
- Revisión y modificación de la documentación del análisis del IDE propuesto según la metodología seleccionada y los posibles cambios surgidos a raíz de las etapas de implementación.
- Entrega de un prototipo funcional.

El presente trabajo está formado por tres capítulos estructurados de la siguiente manera:
Capítulo 1. Se realiza una investigación detallada del estado de arte de las herramientas para la creación de multimedia, además de un resumen de las características de las multimedia. Además se concretan las metodologías, herramientas y tecnologías a utilizar en la propuesta a desarrollar.
Capítulo 2. Se realiza un bosquejo del entorno de desarrollo de la propuesta del sistema, dando al traste con sus requisitos funcionales y no funcionales así como la descripción de los diferentes casos de uso generados durante su análisis.
Capítulo 3. Se desarrolla el diseño de la propuesta partiendo de los patrones que guían su arquitectura. Además se promueve el uso de un manual del desarrollador para futuras mejoras.
Se describe la fase de implementación.

CAPÍTULO 1

“Fundamentación Teórica”

1.1 Introducción

La alta demanda de la producción de multimedia educativa en la UCI así como el proceso de migración a software libre que se usa como política de trabajo, han propiciado un acercamiento entre las comunidades de software libre y software educativo. Para el estudio de una solución que permita a la comunidad de software educativo sumarse al proceso de migración sin influir negativamente en la producción de multimedia este capítulo se centrará en la investigación de herramientas y materiales de libre acceso; así como en la existencia de soluciones que faciliten el desarrollo de un editor visual para el manejo de recursos multimedia.

1.2 Análisis de otras soluciones existentes.

Son pocas las soluciones que se podrían encontrar hasta el momento debido al escaso desarrollo de aplicaciones dentro de la naciente Industria del Software en el país, por lo que este estudio después de una búsqueda exploratoria se ha basado en un grupo de propuestas desarrolladas por equipos de desarrolladores independientes en el mundo. Las candidatas a un análisis riguroso fueron:

F4L

El F4L fue un programa que se desarrolló como una propuesta para Linux del Macromedia Flash, solo que su divulgación no ha sido muy difundida. F4L avanzó mucho en la interfaz de usuario, logrando casi una copia del Macromedia Flash, sin embargo no tiene implementada casi funcionalidades, solo permitía el dibujo de algunas formas sencillas. Contaba con:

- Línea de tiempo.
- Panel de herramientas.
- Biblioteca.
- Panel de propiedades.

Debido a problemas con su diseño este proyecto fue abandonado.

QFlash

QFlash fue una propuesta para realizar un clon de Macromedia Flash para Linux, esto se logró, principalmente, en la parte de la interfaz, que es en gran parte muy parecida a la de Macromedia Flash, pero aún carente de muchas de sus funcionalidades. Aunque QFlash no se terminó, logró desarrollar una interfaz muy parecida a la del Macromedia Flash. Integraba un panel de herramientas, línea de tiempo, inspector de componentes y un editor para Action Script (muy sencillo). Entre los avances de este proyecto se encuentran:

- Generaban películas SWF.
- Permitía el dibujo.
- Trabajo con capas.
- Salvaban en un formato propio el proyecto, para poder editarlo luego.
- Permitía Action Script muy básico.

Más tarde, QFlash se unió con el F4L para crear un programa conjunto: Uira.

UIRA

El proyecto UIRA combina los recursos y conocimiento de los proyectos F4L y QFlash, el mismo pretendió ser un completo IDE como alternativa al software propietario Macromedia Flash MX. UIRA es un acrónimo para UIRA Isn't a Recursive Acronym. Actualmente el proyecto ha sido cancelado debido a la ley DADVSI en Francia (RAFFARIN & AILLAGON, 2003). Se trabajo mucho en el diseño de clases, en el prototipo no funcional y en la organización del proyecto para evitar problemas como en el F4L; sin embargo no se llego a implementar nada, quedándose en la fase de análisis.

KTOON

Ktoon es uno de los programas colombianos, realizado por la empresa Toonka Films, es Software Libre que funciona bajo plataforma Linux, su objetivo no es realizar un clon de Flash, sino un programa que ofreciera la funcionalidad de Flash en el SO Linux. Ktoon es un programa que

permite crear animaciones en 2D del tipo cartoons y Anime, lo que lo hace una aplicación gráfica muy profesional; no solo sirve para hacer la labor de dibujo, sino también el render del mismo. El KToon constituye la alternativa más cercana al Flash pero aunque tiene muchas de sus opciones, cabe señalar que la interfaz del programa es distinta.

Uno de los puntos que se debe destacar con Ktoon es la claridad de los manuales que están en su página, tanto en español como en inglés, los cuales son muy completos y evidencian el poder y lo práctico del programa, una vez que se sabe manejar. Ktoon usa licencia GPL para su uso. Ktoon no implementa un editor de código Action Script ni tiene forma de añadírselo a la película. Por lo que se queda como solo un software para el dibujo de animación.

PENCIL

Pencil 0.4.4b es un software completamente gratis para la animación y el dibujo, que te permite crear los tradicionales dibujos animados utilizando tanto mapas de bits como gráficos de vectores. Es un software específico para animaciones vectoriales orientadas a la web como Flash. Con Pencil 0.4.4b las animaciones pueden exportarse a archivos de tipo Flash y SWF. Sin embargo es un editor gráfico orientado específicamente al diseño y su sistema de animación cuadro por cuadro es totalmente manual.

INSKAPE

Inkscape 0.42 es un programa para crear gráficos vectoriales en 2 dimensiones, capaces de importar formatos como AI, EPS, PNG, TIFF entre otros y exportar PDF y PNG. Cuenta con herramientas para trabajar con texto, trazos, vectorización de mapas de bits, etc. Inkscape es un software de código abierto con capacidades similares a Illustrator, Freehand, CorelDraw o Xara X, que usa como formato nativo el estándar de la W3C basado en XML: Scalable Vector Graphics (SVG). Sin embargo Inkscape no soporta animación SVG, aunque se pueden usar sus gráficos en animación Flash o GIF.

SWFMILL

Es un XML2SWF2XML (OSFLASH, 2007) con importantes funcionalidades. Explicándolo de otra forma es un traductor de XML a un swf y los swf en XML. La sintaxis es fácil y sirve para agregar objetos externos al swf con posterior utilización. Su uso más común es la generación de componentes de la librería conteniendo imágenes (PNG y JPEG), fuentes (TTF) u otras películas swf para su uso con los compiladores de actionscript MTASC (OSFLASH, 2006) o haXe. Además Swfmill puede ser usado para producir estructuras swf tanto complejas como simples.

Fue construido basado en los procesadores XSTL/EXSTL (libxstl). Las entradas o salidas pueden transformarse en xstl que puede ser usado como XML o SWF binario. Usa comandos propios de

XSTL para importar PNG, JPEG, TTF y SWF y para el mapeo de los números identificativos de los SWF. Construido con un lenguaje simple que soporta la creación de bibliotecas y la construcción de simples SWFs. Swfmill es Software Libre, bajo los términos de GNU General Public License.

SWFTools

SWFTools reúne un grupo de herramientas para crear y manipular ficheros swf (BÖHME, 2001). El mismo ha sido liberado bajo licencia GPL, y funciona en entornos Windows, Mac OS X, Linux y otros sistemas tipo Unix. La herramienta principal es SWFC, que recoge la descripción de la animación Flash en un lenguaje sencillo y genera el fichero de salida SWF. Es posible incluir scripts actionscript en el fichero generado. SWFTools también incluye la biblioteca RFXSWF, permitiendo a programas de terceros generar ficheros SWF. Además incluye algunas herramientas para convertir el contenido de formatos JPEG, GIF, WAV y AVI en SWF, así como para extraer el contenido de ficheros SWF.

SWFTools incluye todas las siguientes herramientas:

1. **PDF2SWF** Convertidor de PDF a SWF.
2. **SWFCombine:** Una herramienta para combinar SWFs, insertar uno dentro del otro.
3. **SWFStrings** Explora el SWF en busca de texto.
4. **SWFDump** Imprime varias informaciones sobre un SWF determinado.
5. **JPEG2SWF** Toma un conjunto de fotos y genera una presentación de imágenes (SlideShow) en un SWF.
6. **PNG2SWF** igual que JPEG2SWF, pero solo para PNGs.
7. **GIF2SWF** Convierte GIFs a SWF. Además puede manejar GIF animado.
8. **WAV2SWF** Convierte archivos de audio WAV a SWFs, usando el codificador L.A.M.E. MP3 encoder library.
9. **AVI2SWF** Convierte animaciones AVI en SWF. Soporta compresión Flash MX H.263.
10. **Font2SWF** Convierte archivos de Fuentes (TTF, Type1) a SWF.
11. **SWFBBox** Permite reajustar el área de un SWF.
12. **SWFC** Una herramienta para crear SWF a partir de un script simple.
13. **SWFExtract** Permite extraer Movieclips, Sounds, Imágenes etc. de un archivo SWF.

14. **RFXSWF** Library Una completa librería que puede utilizarse para la generación de SWF, incluye soporte para Bitmaps, Botones, Formas, Sonidos, etc. además soporta actionscript gracias al Compilador Ming.

Dentro de este grupo de herramientas solo constituyen una base para una solución factible el SWFTools y el Swfmill, pues las mismas son capaces de brindar las funcionalidades necesarias para el desarrollo de una multimedia elemental. Cabe señalar que las demás soluciones sirven de guía a la hora de tomar referencias en cuestiones elementales de un IDE tales como diseño de las interfaces y los elementos y requerimientos básicos de este tipo de herramienta.

1.3 La tecnología multimedia.

Multimedia es todo sistema que utiliza más de un medio de comunicación al unísono para la presentación de la información, como la imagen, la animación, el vídeo y el sonido. Este concepto es tan antiguo como la comunicación humana debido a que cuando dos o más personas se comunican en una charla normal hablan (sonido), escriben (texto), se observan mutuamente (vídeo) y accionan con gestos y movimientos de las manos (animación). Con el auge de las aplicaciones multimedia para computador este vocablo entró a formar parte del lenguaje habitual.

Cuando un software, un documento o una presentación combinan adecuadamente los medios, se mejora notablemente la atención, la compresión y el aprendizaje, ya que se acerca un poco más a la manera habitual en que los seres humanos se comunican. La multimedia es el uso de diversos medios (e.g. texto, audio, gráficos, animación, vídeo, e interactividad) de transporte de la información. La multimedia encuentra su uso en varias áreas, incluidas: el arte, la educación, el entretenimiento, la ingeniería, la medicina, las matemáticas, los negocios, y la investigación científica.

En la educación (CARRILLO, 2008), la multimedia se utiliza para producir los cursos de aprendizaje computarizado (CBT) y los libros de consulta como enciclopedias y almanaques. Una enciclopedia electrónica multimedia puede presentar la información de maneras mejores que la enciclopedia tradicional, así que el usuario tiene más diversión y aprende más rápidamente.

La multimedia es muy usada en la industria del entretenimiento, para desarrollar especialmente efectos especiales en películas y la animación para los personajes de caricaturas. Los juegos basados en multimedia son un pasatiempo popular transportado en CD-ROM o disponible en línea. Los usos de la multimedia permiten que los usuarios participen activamente en vez de ser receptores pasivos de la información.

Clasificación y descripción de los recursos editables.

Los recursos multimedia son muy importantes en las animaciones. Los usuarios de herramientas de autor deben poder manejar los recursos que habrán de combinarse en la multimedia resultante, por lo que la mayoría de estas herramientas permiten importar recursos multimedia en diferentes formatos. Uno de los paradigmas de la informática actual es la capacidad de interacción entre diferentes herramientas, por ello es necesario que un programa pueda utilizar los documentos creados por distintas aplicaciones.

Cualquier usuario vinculado a la producción de software educativo en formatos multimedia (DIGIMATCH., 2007) debe conocer las alternativas que tiene a la hora de manejar cualquier recurso multimedia en el ordenador para el desarrollo de la misma. Para sistemas operativos (SO) como Windows existen potentes herramientas propietarias que manejan disímiles formatos, sin embargo esto no ocurre igual para los sistemas derivados de Unix. Por lo que entre las posibilidades que brinda cada herramienta, es imprescindible distinguir los distintos tipos de archivos que puede utilizar, así como sus ventajas y limitaciones.

Recursos gráficos

En el mundo del diseño gráfico (CARREIRA, 2007) existen dos grandes tipos de archivos: mapa de bits y vectoriales.

Las imágenes rasters (mapas de bits):

Los programas de dibujo en mapa de bits (bitmap) funcionan sobre un modelo de píxeles. Una imagen en mapa de bits consiste en una matriz de puntos de mayor o menor resolución espacial y mayor o menor profundidad de color. Además un archivo bitmap debe guardar todos los puntos de la figura para guardar la información de un círculo. Los parámetros que definen la estructura de la imagen en un archivo de mapa de bits son la profundidad de color y la resolución espacial:

El concepto de **profundidad de color** hace referencia a la cantidad de información que se guarda para cada elemento de la imagen (píxel). El concepto de **resolución espacial** solo tiene sentido en las imágenes bitmap, ya que una imagen vectorial únicamente queda limitada en su resolución por la calidad del dispositivo de salida (pantalla, impresora, plotter). La resolución espacial es el número de píxeles que representa la imagen original.

Otro parámetro de interés en los archivos de mapa de bit es el sistema de **compresión** utilizado. Existen dos métodos para reducir el tamaño de estos archivos: métodos con pérdida de calidad y métodos sin pérdida. Las imágenes rasters son muy poco eficientes en su uso de espacio en disco, pero pueden mostrar un buen nivel de calidad. A diferencia de los gráficos vectoriales, al ser reescalados a un tamaño mayor, pierden calidad.

Las imágenes vectoriales:

Los programas de diseño vectorial (LILLEY, 2004) almacenan la información en primitivas gráficas, así para guardar la información de un círculo por ejemplo se guarda la posición del centro, el radio, el tipo, color, grosor de la circunferencia y el relleno. A diferencia de los formatos bitmap no existe un formato tan estandarizado como JPG. Quizás la razón de esta dispersión sea la falta de un formato de gráficos vectoriales para Internet. Existen algunas tentativas en este sentido como es el caso de los gráficos vectoriales de Flash. Aunque hoy para poder producir gráficos de este tipo se necesita la herramienta Flash de Adobe, o de algunas otras como Inkscape (SVG).

Las imágenes vectoriales son más flexibles que las de mapa de bits porque pueden ser redimensionadas y extendidas sin perder calidad. Incluso la animación por gráficos vectoriales suele ser más sencilla y ocupar menos espacio que las de gráficos de mapa de bits. Otra ventaja de los gráficos vectoriales es que su representación suele requerir menos memoria y menos espacio de almacenamiento.

Según las categorías existentes los formatos seleccionados para ser manejados por el editor visual son:

1. Imágenes estáticas de mapas de BIT (JPEG, PNG).

   El formato **JPEG** *(Joint Photographic Experts Group)* es uno de los formatos más utilizados, se trata de un formato bitmap con distintos niveles de compresión con pérdida de calidad. Su principal ventaja es que permite reducir el tamaño de las imágenes. Permite cualquier resolución espacial y cualquier profundidad de bits, por lo que es el más apropiado para la fotografía digital. Los algoritmos de compresión de JPG tienen una gran difusión. JPEG es un algoritmo diseñado para comprimir imágenes con 24 bits de profundidad o en escala de grises.

   El formato **PNG** *(Portable Network Graphics)* representa un intento de estandarizar los formatos gráficos presentes en Internet combinando las posibilidades de transparencia y animación del GIF con las posibilidades de compresión, resolución y colores de JPEG. Es un formato gráfico basado en un algoritmo de compresión sin pérdida para bitmap no sujeto a patentes. Este formato fue en buena parte desarrollado para solventar las deficiencias del formato GIF y permite almacenar imágenes con una mayor profundidad de color. El método de compresión utilizado por el PNG es conocido como deflación.

2. Imágenes estáticas vectoriales (SVG).

   El SVG permite tres tipos de objetos gráficos:

- Formas gráficas de vectores (p.e. caminos consistentes en rectas y curvas, y áreas limitadas por ellos).
- Imágenes de mapa de bits /digitales
- Texto.

Los objetos gráficos pueden ser agrupados, transformados, compuestos en objetos previamente renderizados, y pueden recibir un estilo común. El texto puede estar en cualquier espacio de nombres XML admitido por la aplicación, lo que mejora la posibilidad de búsqueda y la accesibilidad de los gráficos SVG. El mismo es un estándar abierto.

3. Imágenes animadas de mapas de BIT (GIF).

   El formato GIF *(Graphics Interchange Format)* de Compuserve junto al formato PNG ha alcanzado una gran popularidad por ser ambos los más utilizados en Internet. Las imágenes GIF utilizan una tabla de colores para representar la paleta de cada imagen (color indexado) por lo que se limitan a mostrar como máximo 256 colores. Utiliza un algoritmo de compresión sin pérdida. El formato GIF es el preferible si se desea que una parte de la imagen sea transparente o si se desea mostrar pequeñas animaciones.

4. Animaciones vectoriales.

   SWF (URLICH, 2002) es un formato de archivo de gráficos vectoriales creado por la empresa Macromedia, aunque también admite bitmap. Son usados para crear animaciones y gráficos en otros medios y su objetivo principal es obtener archivos pequeños que permitan la interactividad y que funcionen en cualquier plataforma, si bien es cierto que está en formato binario y por lo tanto no es de lectura accesible la especificación completa del formato está disponible. El Plugin que permite reproducir ficheros SWF está disponible en Adobe para diferentes navegadores y diferentes SO.

Recursos de sonido

El sonido al igual que los elementos visuales, tiene que ser grabado y formateado de manera que la computadora pueda manipularlo y usarlo en presentaciones. Existen diferentes tipos de formatos audio, para el IDE en desarrollo se considera en esta parte utilizar formatos propietarios y abiertos, ya que no existe una definición de soporte bien definida en este sentido. Para una primera versión se manipularan solo los formatos wav y mp3.

MP3 (MPEG Audio Layer 3)

El formato MP3 es un formato de compresión de sonido con pérdida (funciona por reducción de datos) recomendado por Moving Pictures Exports Group (MPEG). Actualmente, es muy popular dado que la comunidad Internet lo utiliza para intercambiar archivos de audio. Pese a su carácter

destructivo, conserva una muy buena calidad (calidad de CD o casi de CD, en función de nivel de compresión) y las diferencias sutiles entre un audio de CD original y su copia codificada MP3 son difícilmente apreciables.

WAV (o WAVE)

El formato WAV es un formato de audio digital normalmente sin compresión de datos, propiedad de Microsoft y de IBM, que se utiliza para almacenar sonidos en la computadora. El formato admite archivos mono y estéreo a diversas resoluciones y velocidades de muestreo. Su extensión es .wav y su principal ventaja es que puede ser convertido a otros formatos.

Recursos de video

Existen diferentes tipos de formatos de video, pero es FLV el formato por defecto de los swf para este tipo de recursos. FLV (Flash Video) es un formato de archivo propietario usado para transmitir video sobre internet usando Adobe Flash Player. Los contenidos FLV pueden ser incrustados dentro de archivos SWF. Entre los sitios más notables que utilizan el formato FLV se encuentran YouTube, Google Video, Reuters.com, Yahoo! Video y MySpace. Flash Video puede ser visto en la mayoría de los SO, mediante Adobe Flash Player.

El archivo FLV soporta dos nuevas versiones del llamado códec "screenshare" que es un formato de codificación diseñado para screencasts. Ambos formatos están basados en mapas de bits y pueden tener pérdida al reducir la profundidad de color y están comprimidos usando zlib. El audio en los archivos FLV se encuentra regularmente codificado como MP3. Existen varios reproductores y herramientas que permiten reproducir y crear flv, y que son libres.

1.4 Herramientas y lenguajes a utilizar.

1.4.1 Análisis de librerías libres para la interfaz de usuario.

Qt

Qt es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario (GUI). Fue creada por la compañía noruega Trolltech (Trolltech, 2007). Qt es utilizada en KDE, un entorno de escritorio para sistemas como Linux o FreeBSD. Utiliza el lenguaje de programación C++ de forma nativa y además existen enlaces para C, Python (PyQt), Java (Qt Jambi), Perl (PerlQt) y Ruby (QtRuby) entre otros. La biblioteca cuenta con métodos para acceder a bases de datos mediante SQL, así como uso de XML y para el manejo de otros ficheros, además de estructuras de datos tradicionales.

Las librerías Qt (LARA, 1997) permiten el desarrollo de GUI con una interfaz nativa. Constan de un conjunto de widgets que pueden ser compiladas para diferentes plataformas: Linux, OS X, Windows; así como de otras muchas funcionalidades pensadas sobre todo para ahorrar tiempo y esfuerzo. El uso de esta librería podría permitir la colaboración entre programadores Java y C++ en un mismo proyecto. La gran ventaja es que utilizando esta tecnología la capa de presentación utiliza directamente código nativo, de manera que en principio la eficiencia de este tipo de interfaces debe de ser superior.

En 1992 Qt apareció como una biblioteca desarrollada por Trolltech (en aquel momento "Quasar Technologies") siguiendo un desarrollo basado en el código abierto, pero no libre. Se usó activamente en el desarrollo del escritorio KDE (entre 1996 y 1998), con un notable éxito y rápida expansión. Situación amenazante para el software libre según el proyecto GNU. Para contrarrestar la situación se plantearon dos ambiciosas iniciativas: por un lado el equipo de GNU en 1997 inició el desarrollo del entorno de escritorio GNOME con GTK+ para GNU/Linux (FOUNDATION, 2007). Por otro lado intentan hacer una biblioteca compatible con Qt pero totalmente libre, llamada Harmony.

En noviembre de 1998, anuncian el cambio de licencia de Qt que, a pesar de todo, no contaba con el beneplácito de la Free Software Foundation. El 4 de septiembre de 2000, Trolltech por fin ofrece la versión 2.2 con licencia GPL. Qt cuenta actualmente con un sistema de doble licencia: una GPL para el desarrollo de software de código abierto (open source) y software libre, y otra de pago para el desarrollo de aplicaciones comerciales. Actualmente se encuentra la versión 4 de la biblioteca, y además de las múltiples mejoras, ahora las librerías Qt son también liberadas bajo licencia GPL para Windows.

Qt se encuentra disponible para las siguientes plataformas:

* X11 - Para X Windows System con licencia GPL.

* Mac - Para MacOS X bajo la licencia GPL.

* Windows - Para sistemas Windows con licencia no libre (Solo hasta la versión 3.X, las versiones posteriores a la 4.X ya son GPL)

* PDA - Con licencia GPL.

Actualmente se encuentra en desarrollo QSA (Qt Scripts for Applications), que permite introducir y crear scripts en las aplicaciones creadas con Qt.

GTK+

GTK+ es un grupo importante de bibliotecas o rutinas para el desarrollo de GUI principalmente para los entornos gráficos GNOME, XFCE y ROX de sistemas Linux. Es software libre (bajo la

licencia LGPL), multiplataforma y parte importante del proyecto GNU. Inicialmente fue creado para desarrollar el programa de manejo de imágenes GIMP, sin embargo actualmente es muy usada por muchos otros programas en los sistemas GNU/Linux. GTK+ se ha diseñado para permitir programar con lenguajes como C, C++, Java (Sun), Perl o Python.

Sin embargo GTK+ lleva a cabo la comunicación entre objetos usando los llamados callbacks. Un callback es un puntero a una función, con este mecanismo si se quiere procesar una determinada función cada vez que ocurre un evento en un objeto, lo que se hace es pasar un puntero de otra función a la función deseada y será esta la que se encargue de llamar al callback en el momento apropiado. Este tipo de comunicación tiene el inconveniente de no ser totalmente seguro, es un sistema inflexible y no está orientada a objetos.

Cairo Graphics

Freedesktop, también conocido como fdo, es un proyecto de software libre iniciado por uno de los desarrolladores de GNOME, Havoc Pennington, para la estandarización, integración e interoperabilidad de los distintos escritorios. Freedesktop surgió principalmente como un proyecto entorno al sistema X para la estandarización de los escritorios de Linux con el fin de conseguir que tecnologías existentes estén más integradas, de forma que no haya una total diversidad y separación entre proyectos.

Entre estos proyectos de estandarización nace Cairo, introducida al mundo del Software Libre por Keith Packard, es una biblioteca de gráficos vectoriales en 2D, diseñada para proporcionar imágenes con alta calidad incluidos en el sistema X, OpenGL, buffers de imagen en la memoria y archivos de imagen. Cairo se diseñó para producir una salida idéntica en todos los dispositivos de salida, mientras que aprovecha la aceleración del hardware cuando este está disponible. Cairo es software libre y está disponible para ser redistribuido y/o modificado bajo los términos de la Licencia Pública (LGPL) versión 2.1 o la Mozilla Public Licence (MPL) versión 1.1.

Selección de las librerías a utilizar:

El objetivo de este trabajo es la creación de un IDE para la autoría de interfaces visuales para el desarrollo de multimedia teniendo como base el uso del formato SWF, basando el mismo en la animación e interacción con gráficos vectoriales. Siguiendo los paradigmas de la Programación Orientada a Objetos como base para el desarrollo de aplicaciones y de los procesos de desarrollo llevados a cabo en la universidad, QT4 es la solución a esta problemática

El porqué de la selección de QT4 se debe al hecho de que además de ser una librería publicada bajo la licencia GPL para la creación de software libre, está hecha en C++, aunque da soporte a más de un lenguaje, es multiplataforma, orientada a objetos completamente, contiene una basta

documentación proporcionada tanto por los propios creadores como por grupos afines y presenta un manejo de los gráficos con un rendimiento increíble comparado con otras API de desarrollo antes mencionadas.

Por estas razones y después de un largo estudio comparativo tanto a nivel de pruebas personales como de consulta a especialistas en este campo, no hay divergencia en cuanto a que el desarrollo de un software con un gran manejo de gráficos de forma eficiente, multiplataforma, libre y bien hecho desde todos los puntos de vista debe sustentarse en QT4.

1.4.2 Estudio del lenguaje y herramientas de desarrollo.

Definición del lenguaje de codificación.

C++ es uno de los entornos más serios y poderosos para desarrollar aplicaciones potentes. Desde su versión 4.0 es uno de los más populares y extendidos de todas las plataformas; la posibilidad de realizar operaciones de bajo nivel en un lenguaje de alto nivel fue la característica más importante que lo hizo tan popular (con el C se realizaron muchísimas soluciones que antes hubiesen requerido utilizar Assembler).

Posibilidad de crear una gran cantidad de tipo de aplicaciones: Con C++ se puede realizar casi cualquier tipo de proyecto, desde una aplicación Windows tradicional, hasta un servicio de Windows NT/2000/XP, un control ActiveX, librería DLL/SO, un entorno de escritorio (KDE), juegos, simuladores, plataformas de desarrollo, etc. Se caracteriza por la abstracción, el soporte para la programación orientada a objetos y el uso de plantillas o programación genérica. Por todo esto se puede decir que C++ es un lenguaje multiparadigma que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos.

C++ (Se toma como referencia al lenguaje Java para definir ventajas y desventajas)

Ventajas:

1. C ++ es más rápido que Java. Se dice que Java 1.0 es cerca de 20 veces más lento que C. Sin embargo, Java 1.1 es casi el doble de rápido que Java 1.0, por tanto el código de C compilado se ejecuta diez veces más rápido que los bytecode interpretados de Java.
2. Es una extensión de C. Por eso, muchos programadores encontrarán muy sencilla la transición, ya que podrán seguir haciendo cosas a la antigua usanza.
3. Permite un control de la memoria y una capacidad de programación de bajo nivel impensable en Java.

4. Por ser un lenguaje que trabaja a bajo nivel gestiona mucho mejor el tratamiento de gráficos vectoriales para aplicaciones de este tipo optimizando mucho la calidad sin ser tan costoso en cuanto a recurso de memoria de la máquina.

Desventajas:

1. Para lograr aplicaciones que se ejecuten en varios SO, se requiere de cierto esfuerzo mayor que en Java, ya que Java es un lenguaje interpretado y C++ es compilado.
2. Es una extensión de C. ¿Pero no es una ventaja?, podría llamarse inconveniente, porque bastantes dogmas de la Programación Orientada a Objetos *(POO)* se sacrifican para hacer hueco al C. Java corrige esos problemas.
3. No presenta un toolkit tan rico como el de Java. Java soporta el desarrollo rápido de aplicaciones, y muchas de las tareas de un programador están resueltas en su toolkit.

C++ por ser un lenguaje compilado es más rápido que Java. Es software libre. C++ trabaja a bajo nivel y por tanto gestiona mucho mejor el tratamiento de gráficos vectoriales para aplicaciones de este tipo. Además contiene librerías para realizar aplicaciones con gráficos vectoriales semejantes a la conocida Adobe Flash.

Entorno de desarrollo: KDevelop

KDevelop es un IDE para sistemas Linux y otros sistemas Unix, publicado bajo licencia GPL. A diferencia de muchas otras interfaces de desarrollo, KDevelop no cuenta con un compilador propio, por lo que depende de gcc *(Colección de compiladores GNU)* para producir código binario. KDevelop usa por defecto el editor de textos Kate. Las características que se mencionan a continuación son específicas del entorno de desarrollo:

1. Editor de Código fuente con destacado de sintaxis e indentado automático *(Kate)*.
2. Gestión de diferentes tipos de proyectos, como Automake, qmake (*para proyectos basados en la biblioteca Qt) y Ant (para proyectos basados en Java*).
3. Navegador entre clases de la aplicación.
4. Interfaz para el conjunto de compiladores de GNU.
5. Interfaz para el depurador de GNU.
6. Asistentes para generar y actualizar las definiciones de las clases y la plataforma de la aplicación.
7. Completado automático del código en C y C++.
8. Compatibilidad nativa con Doxygen *(herramienta libre para generar documentación a partir de código fuente).*

9. Permite control de versiones.

La versión de KDevelop a utilizar es la 3.4x o versiones superiores a esta, ya que es la última versión con funcionalidades superiores a versiones previas capaz de soportar Qt4.4.

1. KDevelop puede especificar la dirección de instalación de QT en la creación de un proyecto.
2. Incluye estilos de uso de QT4.4.
3. Presenta nuevas opciones para QT4 en la configuración de proyectos.
4. Integración con Qt Designer.
5. Perfeccionamiento de QMake Manager.

El KDevelop es un IDE para la programación en el entorno KDE, es totalmente gráfico y combinado con el Qt Designer permite realizar aplicaciones en poco tiempo.

Entorno de diseño: Qt Designer

Qt Designer (LARA, 1997) es una aplicación mediante la cual se puede realizar el diseño de aplicaciones GUI de forma gráfica y muy intuitiva usando las librerías Qt. Su facilidad de uso permite en pocos minutos crear los elementos de la interfaz, asignarle los nombres y crear los eventos, así como las funciones que se necesita que realice, para luego codificarlos usando un Lenguaje de POO como es C++.

Características:

1. Dispone de una paleta de widgets muy completa, que incluyen los widgets más comunes de las librerías QT. Si además se instalan las librerías para el desarrollo de aplicaciones KDE, se tendrían widgets adicionales. Las propiedades de un widget cualquiera se pueden cambiar fácilmente en tiempo de diseño con el panel de propiedades.
2. La descripción de la interfaz se guarda en ese fichero en formato XML.
3. Se pueden añadir los siguientes objetos:
   - Botones: Se pueden añadir diversas clases de botones. Los que se pueden encontrar son pushbutton, toolbutton, radiobutton o check button, estos difieren en su forma pero básicamente pueden realizar las mismas funciones.
   - Containers: Se pueden definir como contenedores que se pueden etiquetar con un titulo y pueden contener texto, imágenes incluso otros objetos tales como botones. Igualmente existen varios tipos Groupbox, Buttongroup, Frame, etc.
   - Inputs: Este apartado incluye LineEdit (líneas de edición) pueden servir para recoger

datos que se introducen por teclado, MultiLineEdit (similar), combobox (listas desplegables) y otros como slider, spinbox o dial.

- Displays: En este apartado se incluyen textlabel (etiquetas de texto), pixmapLabel (imágenes), progressbar (barras de progreso) además de LCDnumber, Line etc.
- Views: Son objetos que pueden albergar iconos, también en este apartado se incluyen las tablas.

4. Se pueden realizar señales, conocidas como Slots y Signal:

   Slots y Signal.- Los slots y signals (señales) son un mecanismo de comunicación entre objetos, esta es la principal característica de Qt y es el rasgo que hace distintas las librerías Qt del resto de herramientas para la elaboración de GUI, es un mecanismo de comunicación seguro, flexible y totalmente orientado a objetos y por supuesto implementado en C++.

1.4.3 Metodología y herramientas case a utilizar.

Metodología de desarrollo: RUP

El Proceso Unificado de Desarrollo de Software (RUP) es la metodología a utilizar, en el momento de adoptar esta metodología como estándar se han de considerar unos requisitos deseables partiendo del análisis de criterios de evaluación de la XP y MSF así como del alcance del proyecto software.

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

Inicio, El Objetivo en esta etapa es determinar la visión del proyecto.

Elaboración, En esta etapa el objetivo es determinar la arquitectura óptima.

Construcción, En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.

Transición, El objetivo es llegar a obtener una versión final del proyecto.

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. El ciclo de vida que se desarrolla por cada iteración es llevada bajo dos disciplinas:

Disciplina de Desarrollo:

1. Ingeniería de Negocios: Entendiendo las necesidades del negocio.
2. Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.
3. Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.
4. Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
5. Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado está presente.

Disciplina de Soporte:

1. Configuración y administración del cambio: Guardando todas las versiones del proyecto.
2. Administrando el proyecto: Administrando horarios y recursos.
3. Ambiente: Administrando el ambiente de desarrollo.
4. Distribución: Hacer todo lo necesario para la salida del proyecto.

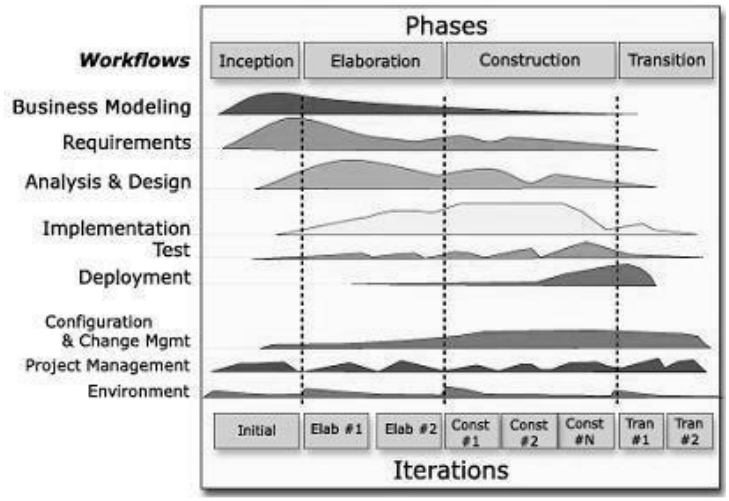


Figura 1.4.1 Fases e iteraciones de la Metodología RUP.

Cada una de estas iteraciones se clasifica y ordena según su prioridad, y luego cada una se convierte en un producto que se entrega al cliente. Como beneficio se puede apreciar la retroalimentación que se tendría en cada versión previa o en cada iteración.

Los elementos del RUP son:

Actividades: Son los procesos que se llegan a determinar en cada iteración.

Trabajadores: Vienen hacer las personas o entes involucrados en cada proceso.

Artefactos: Un artefacto puede ser un documento, un modelo, o un elemento de modelo.

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software.

Existen otras metodologías que son exitosas también como la XP (eXtreme Programing) pero la desventaja que presenta esta metodología es su utilidad en proyectos de corto plazo, corto equipo y cuyo plazo de entrega es muy rápido, además es una metodología de programación rápida o extrema donde cuya particularidad es tener como parte del equipo al usuario final.

El Lenguaje Unificado de Modelado (UML)

Entre los lenguajes de modelado que define OMG *(Object Management Group)* el más conocido y usado es UML *(Unified Modelling Language).* UML es un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan un sistema (LARMAN, 2003). UML fue diseñado para ser un lenguaje de modelado de propósito general, por lo que puede utilizarse para especificar la mayoría de los sistemas basados en objetos o en componentes, y para modelar aplicaciones de muy diversos dominios de aplicación y plataformas de objetos.

En UML 2.0 se definen un conjunto de diagramas organizados en torno a dos categorías: diagramas estructurales y diagramas dinámicos o de comportamiento.

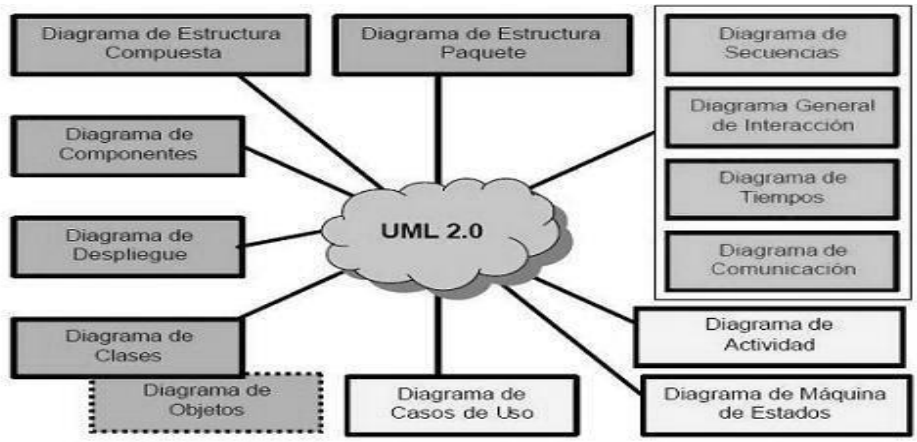


Figura 1.4.2 Diagramas de UML 2.0.

UML 2.0 es la mayor revisión que se le ha hecho a UML desde la versión 1.0. El modelo conceptual en el 2.0 ha sido reestructurado completamente y nuevos diagramas han sido incorporados. Los diagramas tradicionales también han sido mejorados, por ejemplo, los

diagramas de secuencia y de despliegue; en el primero es posible hacer referencias a otros diagramas de secuencia, además de incluir flujos alternos, opcionales y lazos, entre otros; y en el segundo, se colocan sobre los diferentes nodos de la infraestructura de red, a modo de artefactos (elementos físicos simples), los elementos componentes del software.

Herramienta case: Visual Paradigm

Visual Paradigm fácil de usar y de soporte multiplataforma, proporciona excelentes facilidades de interoperabilidad con otras aplicaciones. Visual Paradigm fue creada para el ciclo vital completo del desarrollo del software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación, también proporciona características tales como generación del código, ingeniería inversa y generación de informes.

Está diseñada para usuarios interesados en sistemas de software de gran escala con el uso del acercamiento orientado a objeto, al igual que lo es el RSA, además apoya los estándares más recientes de las notaciones de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros.

Otra herramienta para modelar es el Umbrello UML Modeller que sirve de gran ayuda en el proceso de desarrollo de software usando el estándar UML (Unified Modelling Language), su propósito principal es el de ayudar en el análisis y diseño de los sistemas, sirviendo de poco en las fases del negocio y sistema del RUP, además Umbrello no es lo suficientemente robusto para un desarrollo a grandes escalas. Umbrello a pesar de ser una herramienta que cuenta con licencia GPL no es factible para el desarrollo de un IDE para productos educativos en formato multimedia.

1.5 Conclusiones

Las soluciones de software libre existentes para la creación de multimedia en el proceso educativo no cubren completamente las necesidades de los diseñadores durante el proceso de desarrollo. Es imperante la creación de una solución más avanzada. Como idea inicial se ha tomado una interfaz visual muy parecida al Adobe Flash, para ello es posible reutilizar parte del código de la interfaz de QFlash y Pencil así como tomar pautas en las otras alternativas investigadas. Para el diseño de clases del sistema se puede tomar y analizar el diseño de clases del UIRA.

Para sintetizar el tiempo de desarrollo de la solución se podrían reutilizar herramientas de software libre que faciliten la implementación de ciertas funcionalidades como el dibujado y que puedan integrarse al proyecto como pequeños módulos. El subsistema de gestión visual deberá además permitir la integración de otros subsistemas como podría ser el caso de un editor de código Action Script, una librería de componentes reutilizables, etc.

En este capítulo se ha ofrecido un análisis de las soluciones existentes. Se llego a la conclusión de la ineficiencia de estas soluciones a la hora de producir una multimedia educativa con todas sus funcionalidades, dejando fundamentada la necesidad del desarrollo de una nueva solución para suplir esta necesidad. Estudiado este capítulo se deja orientada la guía de desarrollo del trabajo y sentadas las bases para un posterior análisis de la propuesta del sistema, específicamente del subsistema de edición visual para recursos multimedia.

CAPÍTULO 2

“Presentación de la solución propuesta”

2.1 Introducción

La modelación cognitiva que se expone proporciona un nivel de descripción del dominio que permitirá mediante un diagrama de clases UML mostrar al usuario los principales conceptos que se manejan en el sistema a desarrollar. Esto ayuda a los usuarios, desarrolladores e interesados, a utilizar un vocabulario común para poder entender el contexto en que se emplaza el sistema. Para capturar correctamente los requisitos y poder construir un sistema correcto se necesita tener un firme conocimiento del funcionamiento del objeto de estudio. Estos requisitos estarán presentes durante todo el ciclo de vida del proyecto dejando una traza en cada etapa del mismo.
Se realiza además una descripción de los casos de uso previamente relacionados en un diagrama de UML. De estos surgirán en próximos capítulos las clases del diseño que comparten responsabilidades de acuerdo a las funcionalidades o requerimientos asumidos por el caso de uso del sistema al que respondan. Se cerrará el capítulo con el estudio de factibilidad del sistema propuesto.

2.2 Descripción del entorno de desarrollo de la propuesta del sistema.

La propuesta del sistema estará representada por un modelo de dominio debido al bajo nivel de estructuración que presenta el negocio, y al hecho que este está altamente centrado en herramientas y tecnologías informáticas. Como resolución determinante el IDE debe tener los siguientes componentes:

1. Ventana principal: La barra de programa del margen superior de la pantalla se denominará ventana principal del IDE. Si se cierra, todas las otras ventanas también finalizan su servicio. Ella contiene una barra de menú principal disponible para todas

las órdenes relacionadas con el procesamiento de un proyecto concreto. La carga y almacenamiento de proyectos pertenecen igualmente al menú, así como la presentación u ocultación de las distintas ventanas del entorno de desarrollo y otras.

Menú principal (menús comunes a cualquier aplicación): El menú principal estará compuesto por los menús de, Archivo, Edición, Ayuda.

- Archivo: Permitirá crear nuevos archivos, abrirlos, guardarlos. Destaca la potencia de la utilidad Importar que inserta en la película actual los tipos de archivos (sonidos, vídeo, imágenes e incluso otras animaciones SWF).
- Edición: Es el clásico menú que permitirá Cortar, Copiar, Pegar para recursos; también permitirá personalizar algunas de las opciones más comunes del programa, así como insertar nuevos fotogramas y objetos externos en la película.
- Ayuda: Desde aquí se podrá acceder a toda la ayuda del manual existente.

2. Panel de herramientas: El panel de herramientas acelera las operaciones más comunes del menú principal. Las herramientas de este panel permitirán hacer figuras, además de pintar, seleccionar y modificar ilustraciones.
3. Línea de tiempo: De forma predeterminada, la línea de tiempo aparecerá en la parte superior de la ventana de la aplicación principal, encima del escenario. Podrá adaptarse en cualquier parte del IDE y será posible ocultarla.

   La línea de tiempo será capaz de organizar y controlar el contenido de la película a través del tiempo en fotogramas. Los componentes principales de la línea de tiempo serán los fotogramas clave y la cabeza lectora.
4. Inspector de propiedades: El inspector de propiedades es una ventana desde la cual se podrá ver y modificar la mayoría de las propiedades de los recursos. Muestra la información y la configuración del elemento que está seleccionado, que puede ser un texto, un componente, una forma, un mapa de bits o un video.
5. Escenario: El escenario es el área de trabajo rectangular donde se colocará el contenido gráfico, que incluye, entre otros: gráficos vectoriales, cuadros de texto, botones, clips de vídeo, imágenes de mapa de bits importadas, etc. El escenario del entorno de edición del IDE representará además el espacio rectangular del proyector en la que aparecerá el documento realizado por el IDE durante su reproducción.
6. Panel biblioteca: El panel Biblioteca es donde se guardarán y organizarán los recursos importados por el IDE, tales como gráficos de imágenes de mapa de bits, archivos de sonido y clips de video.

Modelo de Dominio

Diagrama de objetos: Muestra instancias de clases (objetos) con valores en sus atributos y relaciones (Pressman, 2002). El primer acercamiento a la ingeniería de software fue a través de la observación de herramientas privativas dedicadas al diseño de multimedia como es el caso de Adobe Flash en sus diferentes versiones. No se presenta un cliente real o una empresa solicitante con servicios a automatizar por lo que no existe un modelo de negocio previo. Este es el resultado del análisis y las pesquisas:

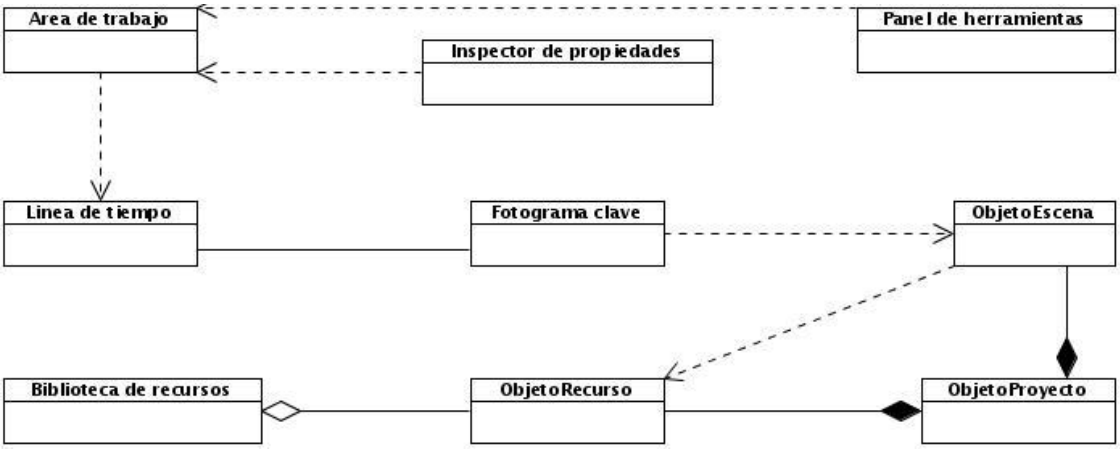


Figura 2.2.1 Diagrama de objetos del Modelo de Dominio.

Glosario de Términos del Dominio

1. Biblioteca de recursos: herramienta en la cual se publicarán y organizarán todos los recursos (solo elementos externos en esta versión) manejados por el entorno de trabajo.
2. Escena: Conjunto de instancias y elementos internos espacialmente dispuestos. Cada escena determina un comportamiento independiente dado por un fotograma clave.
3. Escenario: herramienta en forma rectangular también denominada área de trabajo, donde se colocarán las referencias del contenido de un proyecto durante su edición. Representa el espacio a ocupar por el proyector durante la reproducción del proyecto en cuestión. Su tamaño es personalizable al proyecto en edición.
4. Fotograma clave: Elemento visual de la línea de tiempo, representa la intervención de una nueva representación de objetos en el espacio y en el tiempo. Determina la escena a representar en el área de trabajo o escenario. Puede producir animaciones sin tener que

dibujar cada fotograma por separado ya que responde a un recurso activo: la escena.

5. Inspector de propiedades: herramienta en forma de panel (ventana). Contiene las paletas de propiedades y efectos aplicables a un objeto previamente seleccionado en el escenario.
6. Instancia: Se llama instancia a todo objeto que haga referencia a un elemento externo o componente. Hereda las propiedades y eventos del mismo, aunque externamente pueda simular otro comportamiento. Entre sus propiedades clasifican la identificación del recurso al que responde, así como las coordenadas físicas que ocupa el mismo dentro del escenario.
7. Línea de tiempo: herramienta del entorno de trabajo ubicada en la parte inferior de la pantalla, debajo del escenario. Conjuntamente con el escenario se encarga de la organización espacial y temporal del contenido de un proyecto determinado. Los elementos principales de la línea de tiempo son los fotogramas y la cabeza lectora (encargada de ubicar en tiempo). El espacio u escena estará representado en el área de trabajo según el fotograma indicado.
8. Panel de Herramientas: herramienta que permitirá acelerar las operaciones más comunes del menú principal tales como crear formas, además de seleccionarlas, moverlas, duplicarlas y modificarlas. En ella se podría además configurar los efectos de color para borde o relleno, así como texturas, gradiente, transparencia y otras propiedades de las herramientas con las que trabaja el escenario.
9. Proyecto: fichero externo independiente al entorno de trabajo. Es reconocible por la extensión glm, la cual permite cargarlo al editor visual para su modificación. El contenido se basa en una o varias escenas y uno o varios recursos representados en lenguaje XML.
10. Elemento externo: se entiende por elemento externo todo fichero cuya extensión es reconocida por el entorno de trabajo. Es una categoría de objeto. Estos se clasifican en imagen (.jpg,.png,.svg), animación (.gif,.swf), audio (.mp3, .wav) y vídeo (.flv).
11. Elemento interno: se denomina elemento interno a aquel objeto que es creado en el entorno de trabajo por el panel de herramientas. Tales como texto, trazo y forma.

2.3 Requisitos del Sistema.

Los requisitos del sistema son una parte importante para el desarrollo de una aplicación, constituyen la base para el aseguramiento del correcto funcionamiento, la presencia y por ende la aceptación del software. Los requisitos se clasifican en funcionales y no funcionales:

2.3.1 Requisitos Funcionales

Los requerimientos funcionales serán las capacidades o funcionales que la aplicación debe cumplir, una serie de actividades que serán objeto de automatización más adelante.

1. Crear nuevo proyecto.
2. Modificar el directorio de trabajo.
3. Abrir un proyecto existente.
4. Permitir cerrar un proyecto.
5. Permitir guardar un proyecto.
6. Mostrar opciones de la película.
7. Mostrar nombre de fichero de salida
8. Mostrar la versión de salida swf del proyecto
9. Modificar las opciones de la película.
10. Modificar nombre de la salida swf
11. Modificar versión de salida swf
12. Permitir cargar solo elementos externos soportados.
13. Importar recurso a la biblioteca.
14. Eliminar un recurso de la biblioteca
15. Exportar recurso hacia el escenario.
16. Mostrar propiedades de un objeto.
17. Modificar propiedades de un objeto.
18. Crear fotograma.
19. Eliminar fotograma.
20. Eliminar fotograma clave.
21. Insertar fotograma clave.
22. Convertir fotograma en fotograma clave.
23. Mostrar fotograma activo.
24. Cambiar fotograma activo.
25. Representar contenido de la escena.
26. Eliminar objetos del escenario.
27. Copiar objetos del escenario.
28. Cortar objetos del escenario.
29. Pegar objetos en el escenario.
30. Aplicar transformación libre.
31. Crear objetos en el escenario.
32. Dibujar línea.
33. Dibujar rectángulo.
34. Dibujar cuadrado (Presionar Shift).
35. Dibujar polilínea.
36. Dibujar elipse.
37. Dibujar círculo (Presionar Shift).
38. Dibujar con lápiz.
39. Rellenar figura.
40. Insertar textos.
41. Utilizar la selección libre.

42. Aumentar/disminuir la escena
43. Mostrar las propiedades de las herramientas.
44. Modificar colores de dibujado.
45. Cambiar gradientes.
46. Cambiar texturas.
47. Cambiar tipos fuentes.
48. Cambiar tamaño de fuente.
49. Cambiar tamaños de líneas.
50. Cambiar tipo de línea.
51. Crear historial.
52. Rehacer, Deshacer

2.3.2 Requisitos No Funcionales

Los requerimientos no funcionales identifican un conjunto de propiedades y cualidades que el sistema debe tener, con los requisitos no funcionales se mencionan las características principales que harán al producto atractivo, usable, rápido y confiable.

Confiabilidad

Se debe garantizar que las paradas de mantenimiento no deben interferir en el correcto desempeño del resto del sistema.

Rendimiento

1. Se debe garantizar que las activaciones realizadas sobre la configuración de la ventana principal (despliegues de ventanas, entre otros) sean en menos de (2) segundos.
2. Se debe garantizar que las modificaciones de configuración del área de trabajo sean reflejados en los paneles de propiedades de los objetos.
3. Se debe habilitar el uso de la aplicación para un único desarrollador, o sea solo se tendrá abierto un proyecto a la vez.
4. Se requiere la instalación del Swfmill y el SWFTools para asegurar el buen funcionamiento del sistema.

Soporte y Portabilidad

1. Se debe garantizar el uso de estándares de diseño y arquitectura que faciliten el mantenimiento y ampliación del sistema en caso necesario.
2. Se debe garantizar la compatibilidad del sistema para múltiples plataformas, la familia de los SO GNU/Linux, Windows, Mac OS.
3. Se debe garantizar que un proyecto creado debe ser transparente a la plataforma de trabajo.
4. Se debe garantizar compatibilidad en versiones, donde una composición creada bajo la versión 1 debe poder abrirse transparentemente bajo una versión posterior.

Requerimiento de ayuda y documentación

Se debe garantizar que el IDE posea una ayuda que brinde al desarrollador en dependencia de la acción u objetos seleccionado una guía para trabajar.

Interfaz

1. Se debe garantizar que la interfaz de la aplicación sea lo más atractiva y clara posible para el desarrollador final.
2. Se debe garantizar que la interfaz sea amigable y de fácil comprensión en su funcionamiento permitiendo la utilización del sistema sin mucho entrenamiento.
3. La aplicación debe proveer al desarrollador una gran versatilidad en la presentación de la información en las pantallas, en cuanto a disponibilidad de los recursos y organización de los mismos.
4. El área de trabajo de despliegue propiciada, debe poderse personalizar.
5. Posibilidad de ocultar y mostrar las ventanas de configuración del proyecto en un área determinada; o sea que aunque se oculten éstas se mantengan activas. También debe ofrecerse la posibilidad de cerrarlas.
6. Mostrar el nombre de la aplicación en la parte superior izquierda mediante una barra de título, el nombre del despliegue sobre el cual se trabaja y el camino de la carpeta que contiene el proyecto.
7. En la línea de tiempo debe mostrar un fotograma clave por defecto, así como el escenario y las ventanas comunes.
8. La aplicación debe estar sectorizada en varias áreas funcionales.
    - Para el Menú y las Barras de Herramientas, utilizar la parte superior.
    - Para el panel de controles y dibujos, biblioteca de plantillas de símbolos gráficos, ayuda dinámica, propiedades de cada recurso y componentes de un proyecto, utilizar el lateral derecho.
    - Para el panel de herramientas, utilizar el lateral izquierdo *(Permitir tener un Menú que posea una serie de opciones que garanticen realizar acciones para el manejo y control de los objetos gráficos).*

Requerimientos de licencias y patentes

Se debe garantizar que el sistema será software libre y, por tanto, los componente software que se utilicen también deberán ser libre o por lo menos abiertos.

2.4 Descripción del Sistema Propuesto.

Un sistema de software se crea para servir a sus usuarios. Por lo tanto, para construir un sistema exitoso se debe conocer qué es lo que quieren y necesitan los usuarios prospectos. Aclarar que el término usuario se refiere no solamente a los usuarios humanos, sino a otros sistemas. En este contexto, el término usuario representa algo o alguien que interactúa con el sistema.

Un caso de uso es una pieza en la funcionalidad del sistema que le da al usuario un resultado de valor. Los casos de uso capturan los requerimientos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso el cual describe la funcionalidad completa del sistema. Este modelo reemplaza la tradicional especificación funcional del sistema.

Descripción de los actores

Actor	**Descripción**
Desarrollador	Es el único actor del sistema. Inicializa todas las operaciones que se realizan en el ambiente de trabajo.

Tabla 2.4.1 Actores del Sistema.

2.4.1 Casos de Uso del Sistema

Diagrama de casos de uso: Muestra los escenarios de uso del sistema, incluyendo los roles de los usuarios. Los casos de uso agrupan o fragmentan funcionalidades del sistema de acuerdo a su relación y complejidad; existe una traza directa entre estos y los requisitos funcionales del sistema. Para ver de forma más detallada esta relación así como la descripción completa de cada uno de los casos de uso diríjase al ANEXO A.

CUS1 “Gestionar Proyecto”: se encarga de la creación, salva, apertura, carga, exportación como swf y cierre de un proyecto específico. Maneja el estado por defecto en que debe encontrarse el entorno de trabajo cuando no hay ningún proyecto activo e indica la activación y desactivación de ventanas que componen o responden a la ventana principal.

CUS2 “Gestionar Opciones”: muestra y permite modificar las propiedades del escenario dentro del entorno de trabajo, el cual define el tamaño de las escenas de la película para su exportación, así como la velocidad en fotogramas por segundo y la versión del reproductor. Este caso de uso es consultado por el CUS Gestionar Proyecto para la salva y exportación de cada proyecto.

CUS3 “Gestionar Herramientas”: activa y desactiva herramientas según la selección del desarrollador permitiéndole configurar las propiedades de cada herramienta una vez seleccionada. Este caso de uso es consultado por el CUS Gestionar Escenario para conocer la herramienta activa, así como sus propiedades para su utilización en el escenario.

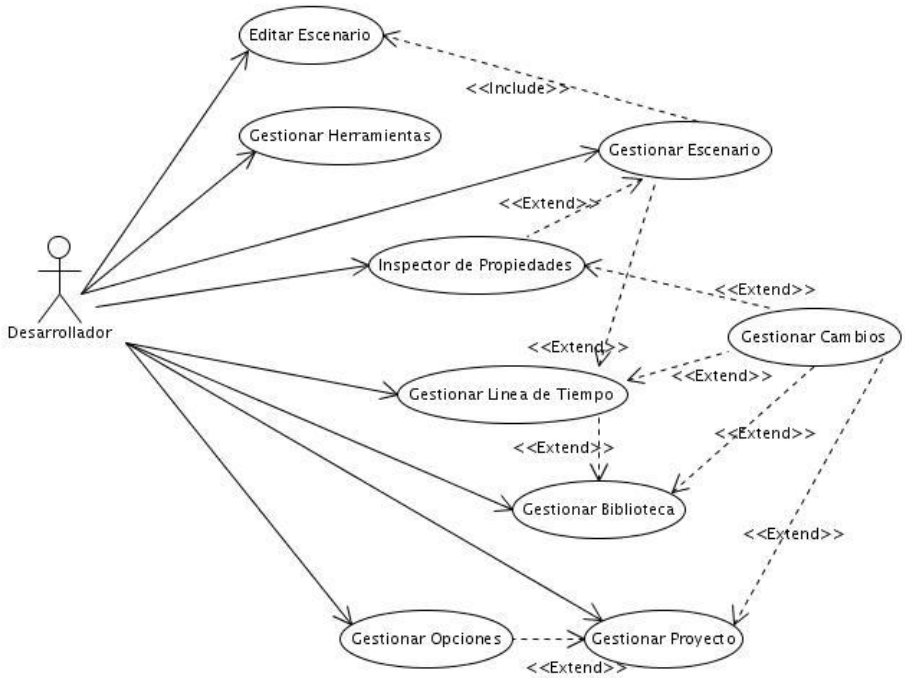


Figura 2.4.2 Diagrama de Casos de Uso del Sistema.

CUS4 “Gestionar Línea de Tiempo”: controla la representación temporal de las escenas del proyecto a través de fotogramas, se encarga de activar o desactivar un fotograma clave ordenando al escenario cargar la escena según corresponda. En su dominio está la creación, eliminación, y transformación de fotogramas y fotogramas clave.

CUS5 “Gestionar Inspector de Propiedades”: es un medio de comunicación entre el desarrollador y los objetos del proyecto, muestra y permite modificar las propiedades del objeto previamente seleccionado por el entorno de trabajo manteniendo actualizada la fuente de datos.

CUS6 “Gestionar Escenario”: se encarga de la representación en el escenario de todos los objetos pertenecientes a la escena correspondiente al fotograma clave activo en la línea de tiempo. Permite editar dicha escena creando nuevos objetos o modificando los existentes. El escenario durante toda su labor consulta al CUS Gestionar herramientas para la creación de nuevos elementos internos y se auxilia del CUS Gestionar Inspector de Propiedades para la visualización de las propiedades del objeto que maneja.

CUS7 “Editar Escenario”: es una pequeña parte del CUS Gestionar Escenario pero es inicializado cuando el desarrollador desea editar un determinado objeto activo en el escenario. El caso de uso se encarga de la representación visual y la salva en el proyecto de las modificaciones que se efectúen sobre el objeto interno en edición. Entre las modificaciones que maneja están la eliminación, corte, copia, pegado y transformación libre del objeto según la opción que se elija.

CUS8 “Gestionar Biblioteca”: maneja los elementos externos asociados al proyecto permitiendo importar nuevos, eliminar los presentes en el proyecto o exportarlos al escenario según el fotograma activo en la línea de tiempo.

CUS9 “Gestionar Cambios”: es accedido desde los CUS Gestionar Inspector de Propiedades, Gestionar Línea de Tiempo y Gestionar Biblioteca ya que el mismo se encarga de registrar todas las acciones efectuadas por el desarrollador para permitirle deshacer o rehacer las mismas, así como facilitarle al CUS Gestionar Proyecto la salva del proyecto activo a partir de las acciones registradas, disminuyendo en gran proporción la cantidad de contenido que debe ser salvado en el fichero físico del proyecto.

2.5 Estudio de factibilidad

Para la realización de un proyecto es importante estimar el esfuerzo humano, el tiempo de desarrollo que se requiere para la ejecución del mismo y también su costo. El estudio de factibilidad del sistema, se efectuará utilizando el método que propone la estimación mediante el análisis de Puntos de Casos de Uso. Se trata de un método de estimación del tiempo de desarrollo de un proyecto mediante la asignación de "pesos" a un cierto número de factores que lo afectan para, finalmente, contabilizar el tiempo total estimado para el proyecto a partir de esos factores. A continuación, se detallan los pasos a seguir para la aplicación de éste método y su resultado.

Identificar los Puntos de casos de uso Desajustados: UUCP = UAW + UUCW

Donde: UUCP: Puntos de Casos de Uso sin ajustar

UAW: Factor de Peso de los Actores sin ajustar

UUCW: Factor de Peso de los Casos de Uso sin ajustar

Para calcular UAW ver ANEXO C Tabla C.1

Para calcular UUCW ver ANEXO C Tabla C.2

Luego: UUCP = 3 + 95 = 98

Ajustar los puntos de casos de uso: UCP = UUCP * TCF * EF

Donde: UCP: Puntos de Casos de Uso ajustados

UUCP: Puntos de Casos de Uso sin ajustar

TCF: Factor de complejidad técnica

EF: Factor de ambiente

Para Calcular TCF

$TCF = 0.6 + 0.01 * \Sigma (Peso_i * Valor_i)$ (Donde Valor es un número del 0 al 5)

Significado de los valores

0: No presente o sin influencia,

1: Influencia incidental o presencia incidental

2: Influencia moderada o presencia moderada

3: Influencia media o presencia media

4: Influencia significativa o presencia significativa

5: Fuerte influencia o fuerte presencia

Ver ANEXO C Tabla C.3

TCF = 0.6 + 0.01 * 25 = 0.85

Para Calcular EF

$TCF = 1.4 – 0.03 * \Sigma (Peso_i * Valor_i)$ (Donde Valor es un numero del 0 al 5)

Ver ANEXO C Tabla C.4

EF = 1.4 - 0.03 * 24 = 0.68

Luego: UCP = 98 * 0.85 * 0.68 = 56.64

Calcular esfuerzo del Flujo de Trabajo de Implementación

E = UCP * CF

Donde: E: Esfuerzo estimado en horas-hombre
UCP: Puntos de Casos de Uso ajustados
CF: Factor de conversión

Para calcular CF

CF = 20 Horas – Hombres (Si Total $_{EF}$ <= 2)
CF = 28 Horas – Hombres (Si Total $_{EF}$ = 3 ó Si Total $_{EF}$ = 4)
CF = abandonar o cambiar proyecto (Si Total $_{EF}$ = 5)

Total $_{EF}$ = Cant EF < 3(entre E1-E6) + Cant EF > 3 (entre E7, E8)

Total $_{EF}$ = 0 + 0 = 0 CF = 20 Horas-Hombres (Porque Total $_{EF}$ < 2)
Luego: E = 56.6 * 20 Horas Hombres = 1132 Horas-Hombres

Calcular esfuerzo de todo el proyecto. Análisis de costos y beneficios

Actividad	%Esfuerzo	Valor Esfuerzo
Diseño	25%	283 Horas-Hombres
Implementación	45%	509.4 Horas-Hombres
Prueba	10%	113,2 Horas-Hombres
Sobrecarga	20%	226.4 Horas-Hombres
Total	100%	1132.0 Horas-Hombres

Tabla 2.5.1 Calcular ET

Como el valor de esfuerzo calculado representa el esfuerzo del Flujo de Trabajo implementación, por comparación salen el resto de los esfuerzo y la suma de ellos es el esfuerzo total (ET).
Si ET = 1132 Horas- Hombre y por cada 140 horas yo tengo 1 mes, eso daría un
ET = 8.00 Mes-Hombre.
Esto quiere decir que 1 persona puede realizar el problema analizado en más o menos 8 meses.

Como el equipo de trabajo cuenta con 2 personas y suponiendo que se realice el mismo esfuerzo la solución propuesta ha de desarrollarse en aproximadamente **4 meses.** El desarrollo de este subsistema (Subsistema de gestión visual) supone 1600 C.U.C en gastos por razones de recursos; que sería el costo aproximado de comprar dos computadoras.

Se debe tener en cuenta el gasto de 2500 pesos cubanos aproximadamente, a incurrir por cuestiones de salario. Se habla de un ahorro de **1700 C.U.C** aproximadamente más un ahorro no calculable por razones de licencia para el desarrollo de este mismo software educativo pero con herramientas privativas. Este subsistema provee grandes beneficios como el hecho de brindar una vía para la migración de la producción de software educativo en formato multimedia al software libre en la UCI.

El desarrollo de esta herramienta tiene un valor social sin precedentes pensando en la necesidad de creación de software educativo hoy en día a nivel nacional y de su influencia en las instituciones de enseñanza desde los niveles más básicos, hasta los sistemas de educación más complejos y especializados como es el caso de las personas con discapacidad.

2.6 Conclusiones

Se ha sintetizado el trabajo abordado a partir de la descripción del modelo de dominio, dejando claros los requisitos funcionales y no funcionales del sistema. En este capítulo se concluyó la descripción de los casos de uso del sistema arrojados por el estudio conjunto del modelo de dominio y los requisitos solicitados. A pesar del gran número de requisitos funcionales y la poca información referente al dominio del negocio, el diagrama de casos de uso representado ha sido desarrollado tomando en cuenta como elemento prioritario la facilidad de mantenimiento y capacidad de reutilización del sistema para el desarrollo de versiones más complejas. El desarrollo del proyecto no solo es factible sino que brinda una ganancia monetaria incalculable y una ventaja legislativa.

CAPÍTULO 3

“Construcción de la solución propuesta”

3.1 Introducción

En este capítulo se modelan los principales artefactos que ayudan a manejar las dificultades que implican la construcción de la aplicación. Para su estudio más detallado se realizarán por cada caso de uso, diagramas de interacción, así como diagramas de clases del diseño. Se abordarán brevemente los patrones de diseño y arquitectura ha utilizar, estándares de codificación, así como los principios del diseño y se finalizará con la representación del modelo de implementación mediante el diagrama de código ejecutable y el diagrama de código fuente que resultó del diseño realizado a cada uno de los casos de uso del sistema.

3.2 Diseño de la aplicación

El diseño es el centro de atención de la propuesta de solución, el mismo contribuye a una arquitectura estable y sólida. En el diseño se modela el sistema y se completa la arquitectura para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Además impone una estructura del sistema que se debe conservar lo más fielmente posible. Previo a realizar los diagramas que muestran la solución del sistema, es necesario definir algunos conceptos que ayudan a alcanzar un producto con una calidad superior.

Diagrama de Clases del Diseño

Diagrama de clases: Presenta las clases, junto con sus atributos, operaciones, interfaces y relaciones. También presenta el agrupamiento de clases en paquetes y las relaciones entre ellos.

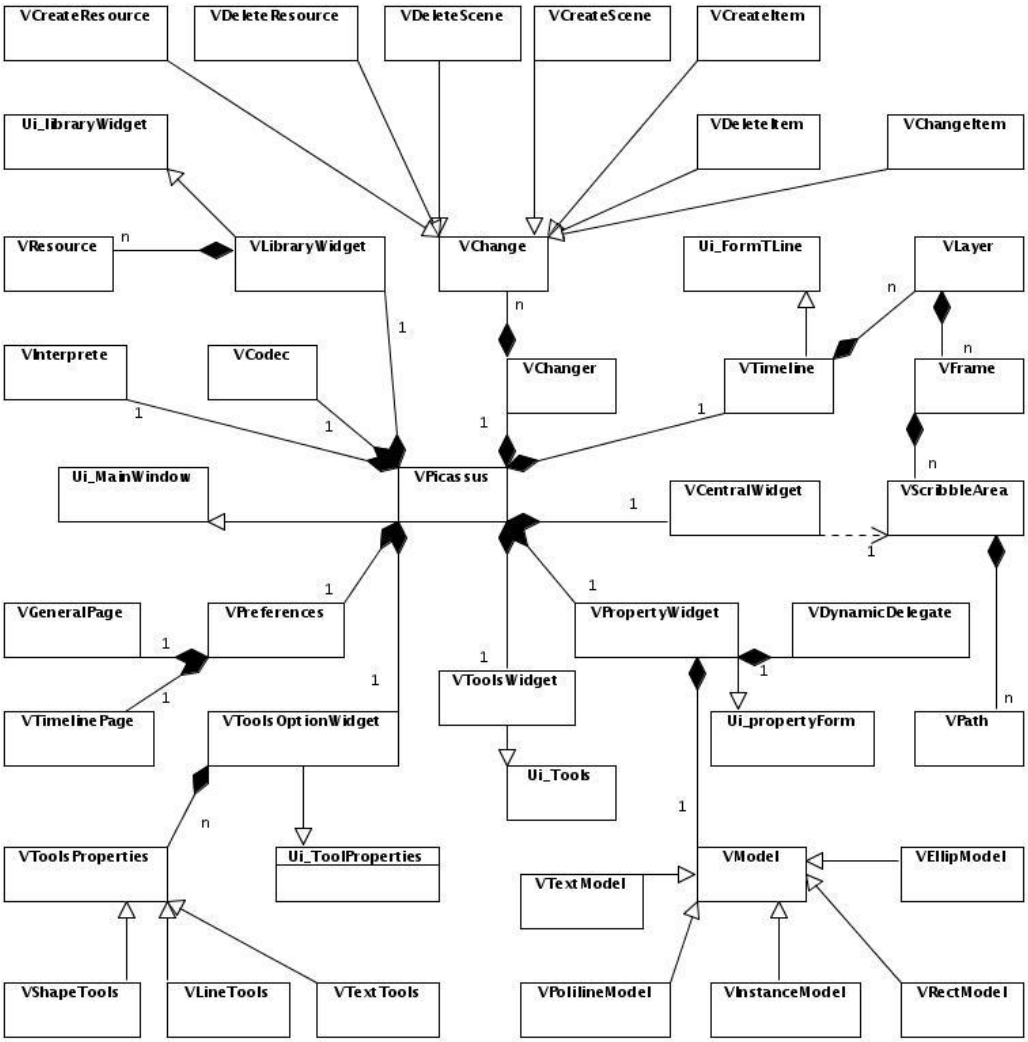


Figura 3.2 Diagrama de Clases del Diseño.

Diagramas de interacción: Comprende los diagramas de secuencia y de colaboración. Presenta objetos y relaciones entre ellos desde el punto de vista dinámico. Estos diagramas estarán disponibles solo desde el expediente del proyecto.

Arquitectura del Sistema

“… La arquitectura de un sistema constituye un amplio marco que describe su forma y su estructura, sus componentes y como estos interactúan…” Jerrold Grochow.

El conjunto de modelos que describen la Línea Base de la Arquitectura se denomina Descripción de la Arquitectura. El papel de la descripción de la arquitectura es guiar al equipo de desarrollo a través del ciclo de vida del sistema. Esta descripción puede adoptar diferentes formas, puede ser un extracto (un conjunto de vistas), o puede ser una reescritura de los extractos de forma que sea más fácil leerlos. Estas vistas incluyen elementos arquitectónicamente significativos (casos de uso, subsistemas, interfaces, algunas clases y componentes, nodos y colaboraciones).
La descripción de la arquitectura tiene cinco secciones, una para cada modelo: una vista del modelo de casos de uso, una vista del modelo de análisis (opcional), una vista del modelo de diseño, una vista del modelo de despliegue (no requerido aplicaciones de escritorio), y una vista del modelo de implementación. Para ver la Descripción de la Arquitectura debe solicitar el expediente del proyecto.
La información brindada en este documento sobre la arquitectura del sistema se concentra en el patrón utilizado para su desarrollo.
El sistema se basa en una arquitectura de N capas organizadas jerárquicamente donde cada capa provee servicios a la capa superior y es servido por la capa inferior. Sus componentes son cada una de las capas y los conectores los protocolos de interacción entre las mismas. Tiene como restricción que la interacción está limitada a las capas adyacentes.

La **capa de presentación** maneja la interacción entre el usuario y la aplicación; contiene la implementación de las interfaces y sus clases controladoras exponiendo de forma amigable y eficiente las funcionalidades que presenta la aplicación tales como la visualización de la información contenida en la escena activa y el listado de recursos almacenados en la librería.

La **capa lógica** de negocio expone las funcionalidades básicas requeridas por la aplicación hacia la capa de presentación. Contiene la implementación de los casos de uso que dan respuesta a los requisitos funcionales del sistema. Es lo que se conoce como la manera en que los datos son persistidos o almacenados en la capa de datos y por tanto implementa el acceso a los mismos.

La **capa de datos** contiene los medios de almacenamiento utilizados por la aplicación, en los que se encontrarán todos los datos referentes a cada proyecto en específico. Esta capa cuenta con un único fichero de extensión dtd para el aseguramiento de que todos los medios de almacenamiento basados en XML cumplan con la debida estructura; y dos ficheros por cada proyecto desarrollado:

uno de extensión glm escrito en lenguaje XML que facilita la comunicación de la aplicación con el contenido del proyecto en desarrollo; y otro con extensión swf que permite la visualización del proyecto por alguno de los reproductores asociados a este tipo de multimedia ya que la aplicación carece de reproductor propio.

Figura 3.3 Arquitectura de N capas

Ventajas de un sistema basado en capas:

- Facilita la descomposición del problema en varios niveles de abstracción.
- Soporta fácilmente la evolución del sistema; los cambios sólo afectan a las capas vecinas.
- Se pueden cambiar las implementaciones respetando las interfaces con las capas adyacentes.

Desventajas del uso de capas:

- No todos los sistemas pueden estructurarse en capas.
- A menudo es difícil encontrar la separación en capas adecuada.

Para la descripción de la arquitectura se ha usado el UML que aunque difiere sustancialmente de los tradicionales lenguajes de descripción de arquitectura (LDA 0 ADL en inglés), permite modelar de forma general la arquitectura de software. No obstante vale destacar que existen herramientas donde se hace uso de los LDA para la verificación de la arquitectura y la obtención de prototipos rápidos.

Descripción de la arquitectura del sistema:

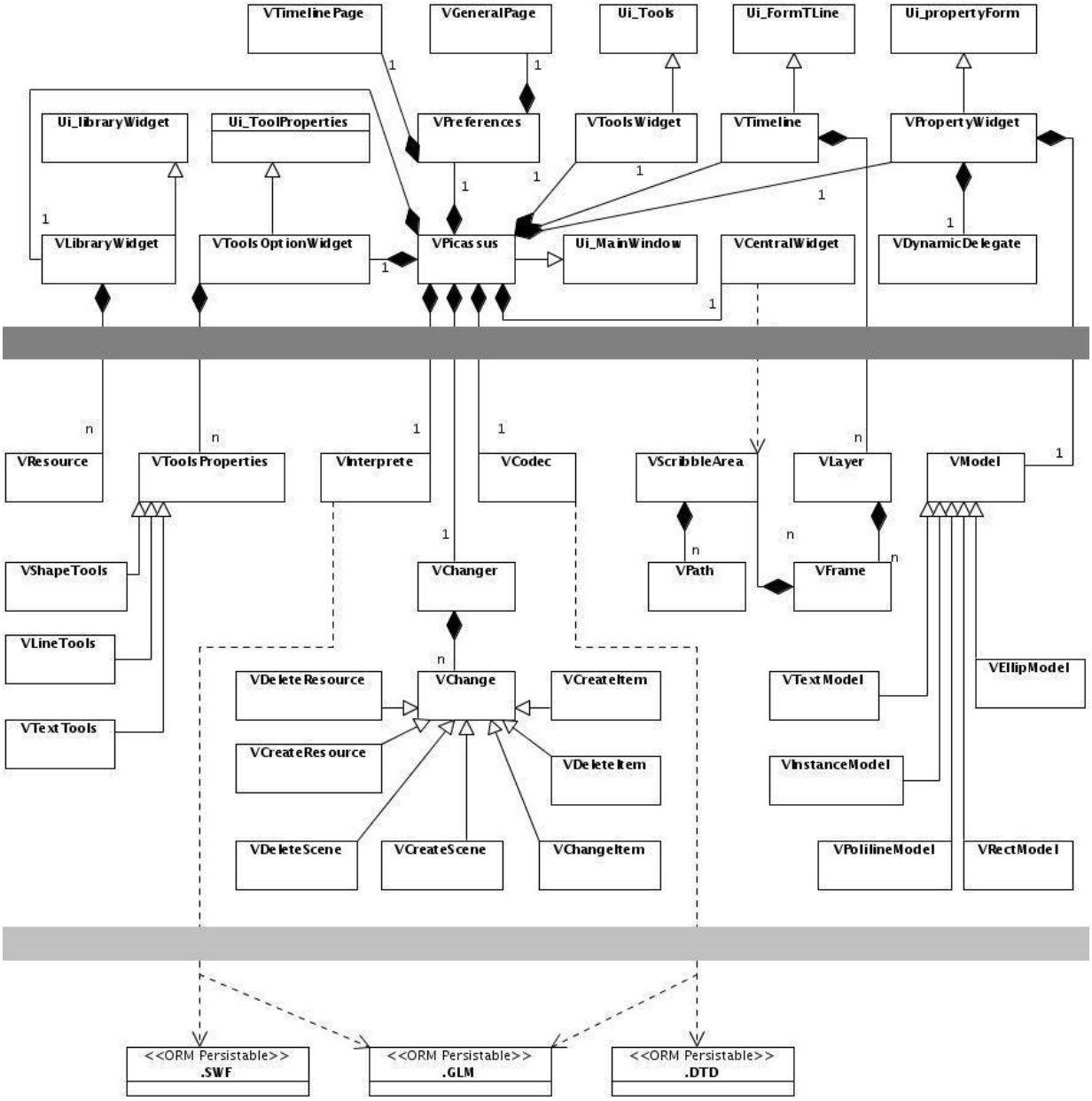


Figura 3.4 Arquitectura del sistema.

3.3 Principios de diseño

Se podrían abordar disímiles principios del diseño sin embargo el más arraigado es el uso de patrones. Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características.

Una de las características que debe tener un patrón es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (Erich Gamma, 1995). No es obligatorio utilizar los patrones siempre, solo en el caso de tener el mismo problema o similar que soluciona el patrón, siempre teniendo en cuenta que en un caso particular puede no ser aplicable. Abusar o forzar el uso de los patrones puede ser un error.

Estándares en la Interfaz de la Aplicación

La calidad de la interfaz de usuario puede ser uno de los motivos que conduzca a un sistema al éxito o al fracaso, es por ello que su diseño es uno de los puntos fundamentales a tratar a la hora de confeccionar la aplicación. Toda Interfaz debe ser lo más amigable y comprensible posible facilitando así su interacción con el desarrollador. Una aplicación con una interfaz bien diseñada debe tener, además de un buen diseño gráfico, una correcta distribución de los contenidos según su funcionalidad.

Para lograr un diseño amigable de la interfaz de la aplicación, se le facilitó al usuario la personalización del ambiente de trabajo, dejando un margen de selección para la localización y visibilidad de las herramientas (ventanas funcionales) del IDE. Se siguieron algunas de las características visuales propias de herramientas privativas como Adobe Flash ya arraigadas en los usuarios y se incorporaron otras. Algunas de las características generales son:

1. Utilizar una misma tipografía, forma y estilo en todas las ventanas.
2. Despliegue visual de mensajes informativos sobre el estado del entorno de desarrollo o sus funcionalidades.
3. Usar iconos comunes para las funcionalidades y los elementos que se repiten en varias pantallas permitiéndole al desarrollador acostumbrarse al ambiente sin dificultades.

4. Representación lineal de los recursos del proyecto a manejar por la interfaz de usuario (Línea de Tiempo y Biblioteca de recursos).
5. Agrupamiento de los medios de edición de objetos que provee la interfaz de usuario (Panel de Herramientas e Inspector de Propiedades).
6. Centralización de la comunicación entre recursos y medios de edición a través de una ventana central que figura como zona de representación y edición.

Los prototipos de las interfaces gráficas pueden observarse en el ANEXO B.

Tratamiento de excepciones

El tratamiento de errores posibilita el buen funcionamiento de una aplicación dándole una mejor apariencia ante los clientes. Para prevenir errores en la aplicación a la hora de efectuar cualquier operación al desarrollador sólo se le brindan las opciones mínimas necesarias, o sea, se deshabilitan determinados botones y se le muestran solo opciones válidas y existentes. Se garantiza además la consistencia e integridad de los datos mediante la validación de funcionalidades y propiedades en tiempo de ejecución. Para esto, se verifican los campos obligatorios y se asignan valores por defecto, siguiendo ciertas reglas de comportamiento.

Estándares de codificación

En general el código resultante debe exhibirse con las siguientes cualidades:

1. Confiabilidad: el producto resultante debe realizar lo que se espera de él con total certeza.
2. Portabilidad: el código debe ser portable a varias arquitecturas y compiladores.
3. Mantenibilidad: se debe escribir el código de manera legible, consistente, con un diseño sencillo y fácil de depurar.
4. Fácil de probar: se debe escribir código fácil de probar, minimizando las siguientes características para lograr que cada módulo sea mantenible y sencillo de probar:
   - tamaño del código
   - complejidad
   - cantidad de parches
5. Reusabilidad: los componentes producidos deben ser reutilizables, así se disminuye la codificación de otros componentes y la prueba, con lo que se reducen los esfuerzos.

6. Extensibilidad: los requerimientos deben cubrirse al 100%, pero el código debe estar listo para asumir los cambios con la mayor prontitud posible, sin provocar que se itere nuevamente todo el ciclo de desarrollo.

Para lograr todas estas características se definieron las siguientes reglas:

Regla 1: Todo el código que se escriba debe cumplir con la norma ISO/IEC 14882:2002(E) referente a los estándares del lenguaje de programación C++.

Regla 2: Ninguna función o método contendrá más de 200 LSLOC (Logical Source Lines Of Code) *Las funciones muy grandes tienden a ser complicadas de leer, entender y probar.

Regla 3: Toda expresión o sentencia se escribirá en una línea independiente.

Regla 4: Todos los identificadores de variables y los valores de enumeraciones se escribirán con letras minúsculas.

const uint16 max_pressure = *100*;

enum switch_position { up, down };

Regla 5: Los identificadores NO comenzarán con el símbolo de subrayado.

Regla 6: Todas las siglas presentes en un identificador se escribirán con mayúsculas.

Regla 7: La declaración de clases que solo se acceden a través de referencias (&) o punteros (*) se debe hacer en archivos de cabecera de “adelanto” que solo contienen declaraciones de clases de “adelanto” (forward declarations).

class *VForward;*

VForward obj;*

Regla 8: Todo archivo de implementación debe incluir un archivo de cabecera que solo contiene definiciones de tipos de datos, funciones inline y definición de plantillas de tipos de datos.

Regla 9: Todo identificador tendrá un sufijo que identifique a que pertenece:

Clases “V” Ej.: ***class*** VRectangle {/*...*/};

Componentes de la interfaz “te (TextEdit), cb (ComboBox), lv (ListView)

Enumeraciones “E”

Regla 10: El nombre de un archivo de cabecera debe reflejar la entidad lógica que define.

Header Matrix.h defines:

class *Vmatrix { /*...*/};*

Regla 11: La interfaz de una clase debe ser mínima y completa.

Regla 12: Las referencias a objetos se especifican al lado del tipo y no del identificador.

int& reference; // Bien, fácil de leer el identificador

float** pointer; // Bien, fácil de leer el identificador

double *other_pointer; // Mal! Difícil de leer.

Regla 13: Toda función miembro, declaración de clases o enumeración, así como los atributos públicos deben ser comentaríados con un resumen de su objetivo y funcionalidad siguiendo la estructura de comentario de Doxygen

/**

* @**class** ***VRectangle*** Clase que me almacenará los datos de un rectángulo

*/

class VRectangle {/*...*/ };

/**

* Función para adicionar elementos a la biblioteca

* @**param** ***nuevo*** Es un nuevo recurso que se desea adicionar a la biblioteca

*/

void VPicassus::addResource (***const*** VResource& nuevo) { /*...*/ }

Regla 14: Cada instrucción if, else, while, do, for, entre otras llevará SIEMPRE asociado un bloque de instrucciones entre llaves.

*Aumenta la legibilidad y se escribirán las llaves aunque se tenga solo una instrucción.

Estas son solo las principales reglas para la codificación del sistema, existen muchas más que ayudan y complementan a estas para cumplir los objetivos del desarrollo de productos con calidad.

Patrones GRASP (Patrones de Software para la Asignación General de Responsabilidad)

Se ha hecho uso en el diseño del sistema de los patrones GRASP para la asignación de responsabilidades, ya que estos describen los principios fundamentales de diseño de objetos para dicha actividad. Los patrones GRASP constituyen además un apoyo para la enseñanza y entendimiento del diseño de objetos, ejerciendo el razonamiento de una forma sistemática, racional y explicable.

Los patrones GRASP utilizados son:

Nombre del patrón	Problema	Solución
Expert - Experto	¿Cuál es un principio general para asignar responsabilidades a los objetos?	Asignar una responsabilidad al experto en información (la clase que tiene la información necesaria para la realización de la asignación).
Creator-Creador	¿Quién debería ser el responsable de la creación de una nueva instancia de alguna clase?	Asignar a la clase B la responsabilidad de crear una instancia de clase A si se cumple uno o más de los casos siguientes: 1. B agrega objetos de A 2. B contiene objetos de A 3. B registra instancias de objetos de A 4. B utiliza más estrechamente objetos de A. 5. B tiene datos de inicialización que se pasarán a un objeto de A cuando sea creado (por tanto, B es un Experto con respecto a la creación de A). 6. B es un creador de los objetos A.
Low Coupling - Bajo Acoplamiento	¿Cómo soportar bajas dependencias, bajo impacto del cambio e incremento de la reutilización?	Asignar una responsabilidad de manera que el acoplamiento permanezca bajo.

High Cohesion - Alta cohesión	¿Cómo mantener la complejidad manejable?	Asignar una responsabilidad de manera que la cohesión permanezca alta.
Controller - Controlador	¿Quién debería ser el responsable de gestionar un evento de entrada al sistema?	Asignar una responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase que representa una de las opciones siguientes: 1. Representa el sistema global, dispositivo o subsistema. 2. Representa un caso de uso en el que tiene lugar el evento del sistema a menudo denominado <nombre del caso de uso> Manejador, <nombre del caso de uso> coordinador, <nombre del caso de uso> Sesión. Utilice la misma clase controlador para todos los eventos del sistema en el mismo escenario de caso de uso. Informalmente, una sesión es una instancia de una conversación con un actor. Las sesiones pueden tener cualquier duración, pero se organizan a menudo en función de casos de uso.

Tabla 3.1 Patrones GRASP

Otros patrones de diseño

Patrones de creación

Los patrones creacionales abstraen el proceso de instanciación, encapsulan el conocimiento sobre clases concretas usadas por el sistema, además de ocultar la forma en que se crean y ponen en contacto las instancias. No son siempre excluyentes, en ocasiones se complementan.

Singleton: El patrón singleton asegura que sólo se pueda crear una instancia de una clase, y ofrece un punto de acceso global a ella.

1. Aplicación

   Existe la necesidad de que una clase se instancie una sola vez de modo que todos los clientes puedan acceder a esa única instancia desde un punto conocido. La instancia única se puede ampliar mediante subclases y los clientes deben ser capaces de utilizar las instancias de las subclases sin tener que modificar su código.

2. Participantes

   Singleton es el responsable de la creación de su única instancia. Define una operación para crear instancias que ofrece a todos los clientes la misma y única instancia. Este método debe ser estático.

3. Representación

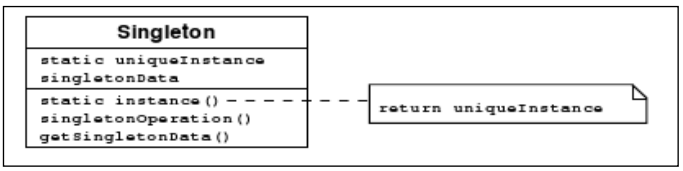


Figura 3.5 Patrón Singleton

Patrones de estructura

Los patrones estructurales separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes. Usan la herencia para el trabajo con clases, de lo contrario describen formas de ensamblar objetos.

Adapter: El patrón adapter convierte la interfaz de una clase en la interfaz que el cliente espera. El adapter permite que clases con interfaces incompatibles puedan trabajar juntas.

1. Aplicación

   Se desea utilizar un clase ya existente pero cuya interfaz no coincide con la que se necesita, por lo que se crea una clase que colabora con otras clases que no tienen interfaces compatibles; o bien se desean adaptar varias subclases ya existentes adaptando la interfaz de su clase padre común (object adapter).

2. Participantes

   - Target: Define la interfaz específica del dominio en el que se quiere hacer uso de la clase que se adapta.
   - Client: Utiliza objetos que implementan la interfaz definida por el target.
   - Adaptee: Presenta su interfaz original, que es la que se tiene que adaptar.
   - Adapter: Adapta la interfaz del objeto adaptado a la definida por el target.

3. Representación

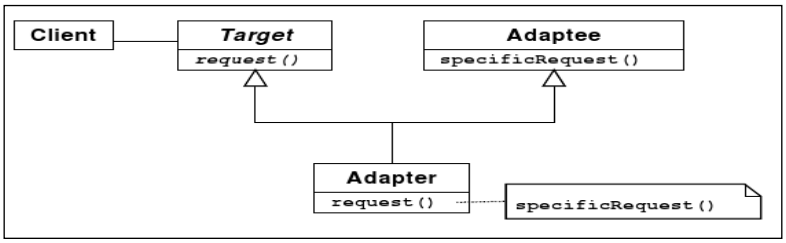


Figura 3.6 Patrón Adapter

Patrones de comportamiento

Los patrones de comportamiento más que describir objetos o clases, describen la comunicación entre ellos. Caracterizan el modo en que las clases y objetos interactúan y se reparten las responsabilidades. Los de clases usan la herencia para describir algoritmos y flujos de control, mientras que los de objetos describen cómo cooperan entre sí un grupo de objetos para realizar una tarea que ninguno podría llevar a cabo por sí solo.

Command: El patrón command encapsula peticiones en forma de objetos permitiendo así parametrizar los clientes utilizando distintas peticiones, encolar las peticiones y ofrecer la posibilidad de deshacer las operaciones. Permite solicitar operaciones sin tener que saber cómo o quién lleva a cabo esas operaciones.

1. Aplicación

   Parametrizar objetos mediante una acción. Especificar, encolar y ejecutar operaciones en distintos momentos. Soportar deshacer operaciones. Soportar restaurar el estado a partir de un momento dado. Ofrecer una interfaz común que permita invocar las acciones de forma uniforme y extender el sistema con nuevas acciones de forma sencilla.

2. Participantes

   - Command: Declara una interfaz de ejecución de operaciones.
   - Concrete Command: Implementa la interfaz de ejecución utilizando las operaciones del receptor, para lo que debe establecer la unión con dicho receptor.
   - Client: Crea un comando concreto y establece quién es el receptor de la acción del comando.
   - Invoker: Almacena un comando concreto y, llegado el momento, le solicita que se ejecute.
   - Receiver: Sabe cómo llevar a cabo las operaciones asociadas a la solicitud. Cualquier clase puede actuar como receptor.

3. Representación

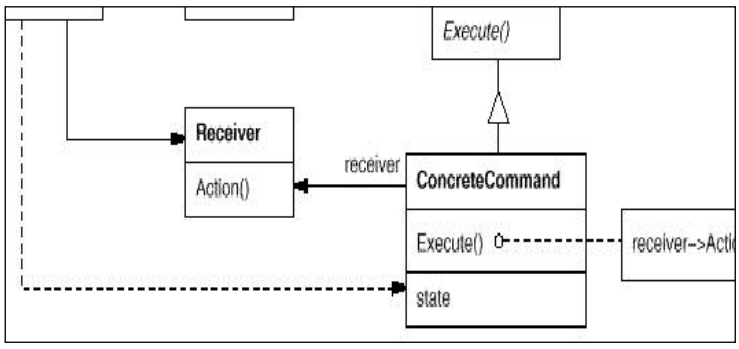
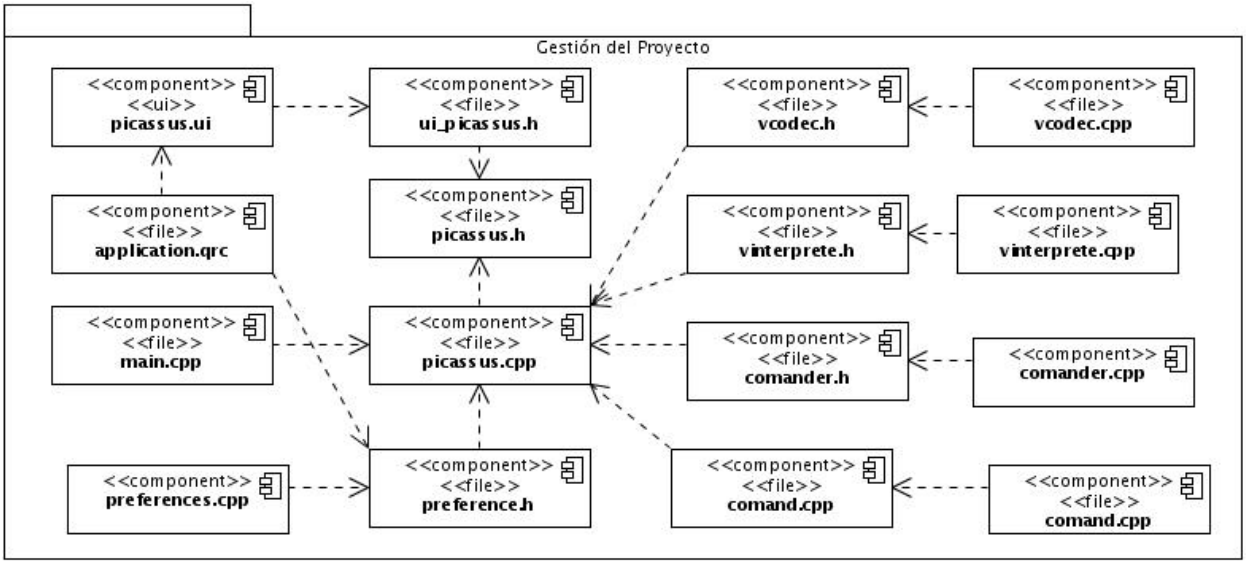


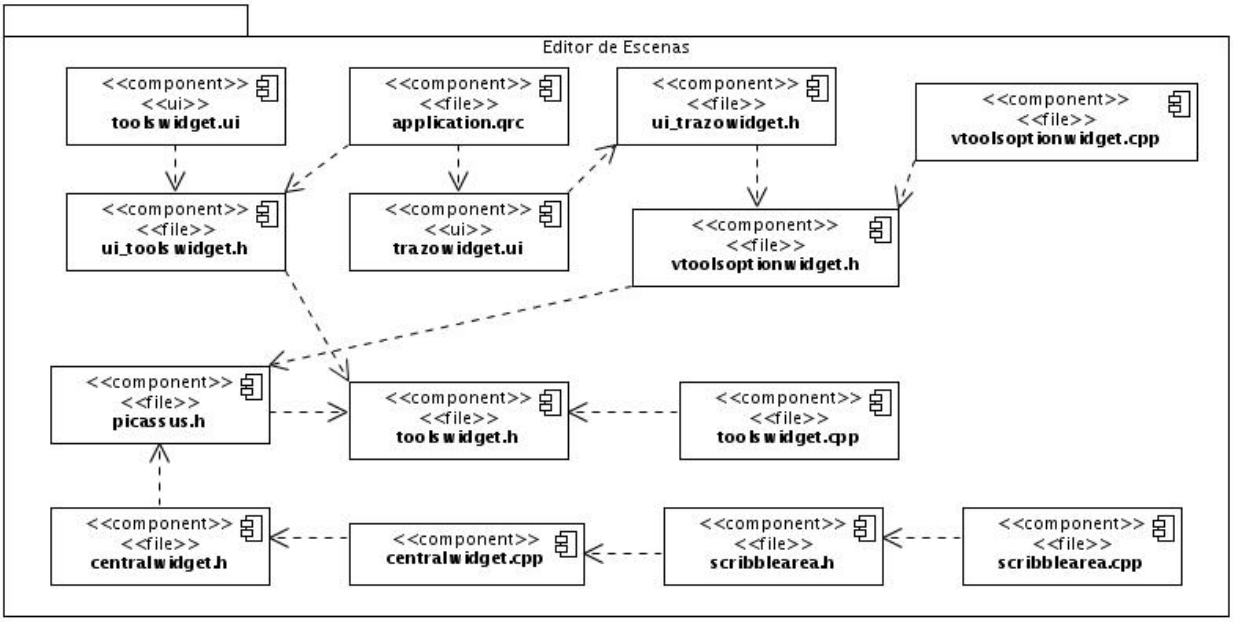
Figura 3.9 Patrón Command

3.4 Modelo de Implementación

Diagrama de componentes de código fuente:

Los diagramas de componentes de código fuente representan dependencias entre ficheros de código fuente, modelan las diferentes versiones de estos ficheros y guían las actividades de implementación.





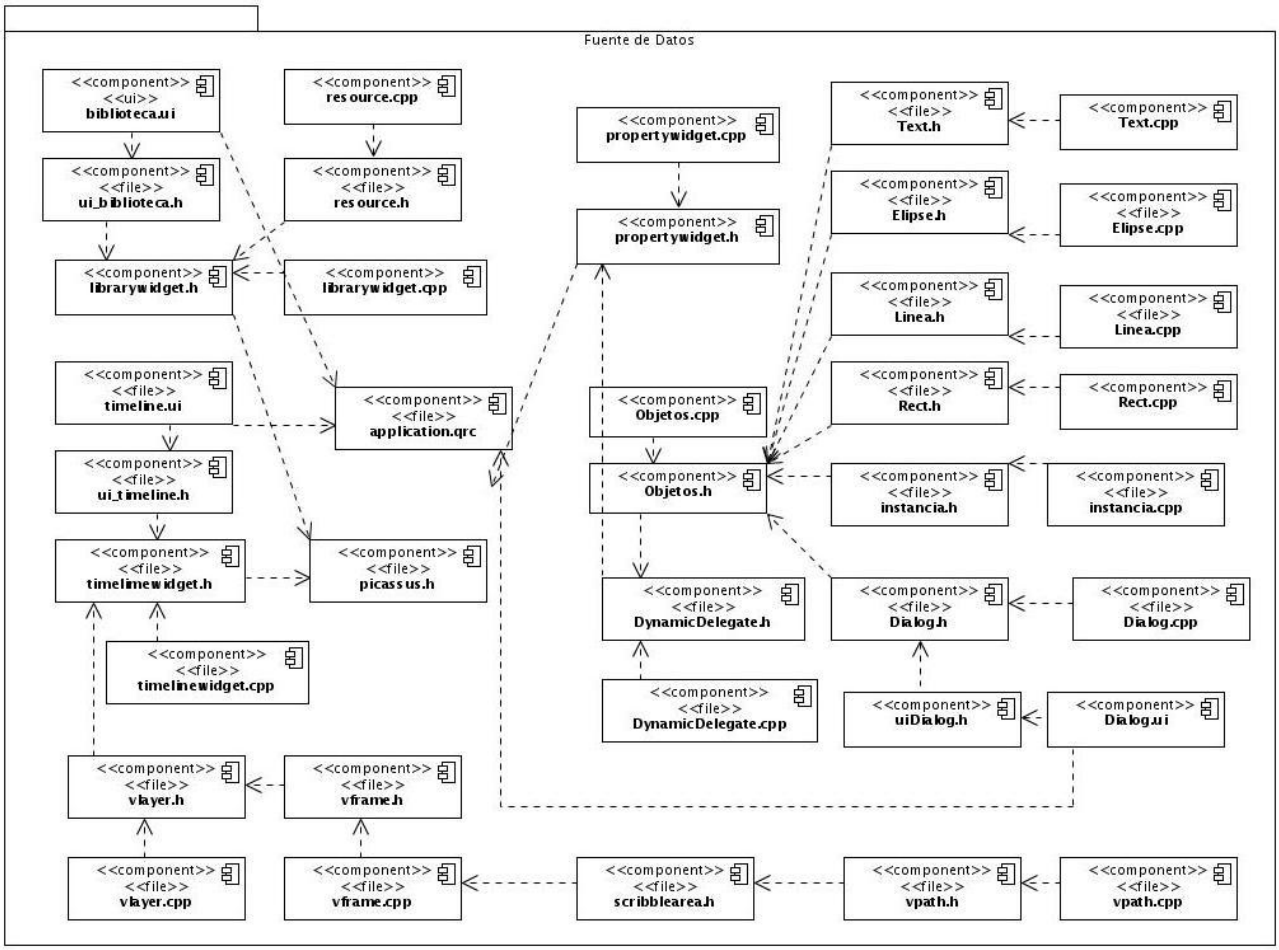


Figura 3.10 Diagrama de componentes Código fuente

Diagrama de componentes de código ejecutable:

Los diagramas de componentes de código ejecutable modelan la distribución de una nueva versión a los usuarios e indican al cliente los componentes físicos con los que debe contar para utilizar completamente la aplicación. En el caso de este sistema se requiere de la presencia de herramientas tales como el SWFC y el SWFMill estudiadas anteriormente en el capítulo 1, así como de las clases que proporciona la librería Qt 4.4. Se representan a su vez en el diagrama ficheros temporales que se requieren para la correcta comunicación entre estas herramientas además del fichero de configuración del proyecto desarrollado y su versión exportable o swf, el cuál requerirá de un reproductor como Gnash para su visualización.

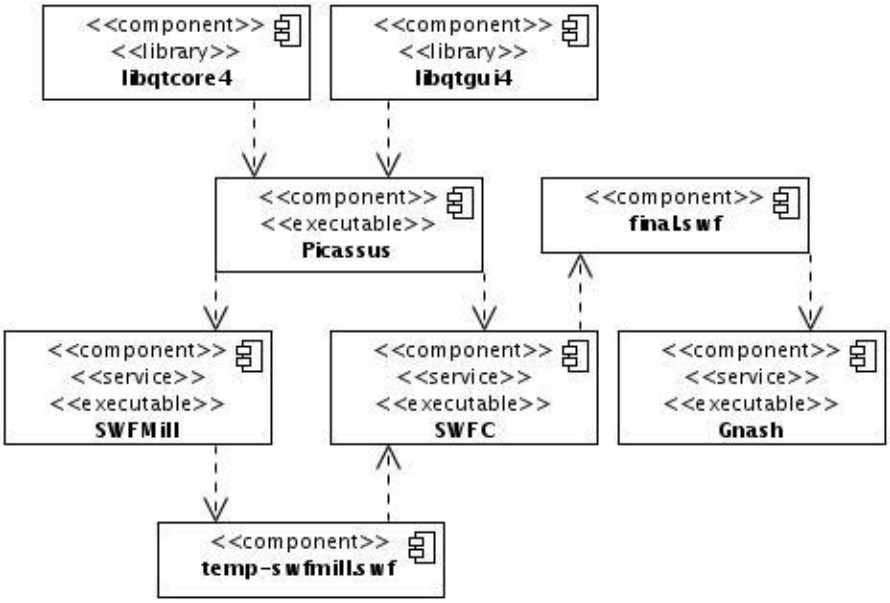


Figura 3.11 Diagrama de componentes Código Ejecutable.

3.5 Concepción general de la ayuda

Por el momento solo se cuenta con una ayuda para desarrolladores provista por el uso de la herramienta Doxygen. El principal interés de esta herramienta es permitir la integración de la documentación directamente en el código fuente, favoreciendo así la coherencia entre ambos. Este software permitió extraer la documentación a partir de un código fuente no documentado de antemano, haciendo uso de ciertos estándares para la escritura de los comentarios.

Parte de la información incluida en la ayuda y partiendo del código fuente es:

1. Prototipo y documentación de las funciones, ya sean locales, privadas o públicas.
2. Lista de archivos incluidos.
3. Documentación de las estructuras de datos.
4. Prototipo y documentación de las clases y su jerarquía.
5. Diferentes tipos de gráficas: diagramas de clase, de colaboración, de llamadas, etc.…
6. El índice de todos los identificadores.

3.6 Conclusiones

En este capítulo se describió detalladamente la solución propuesta a través del modelo de diseño y de implementación. Se describió exitosamente la arquitectura en N capas y se expusieron los patrones de diseño y arquitectura utilizados en la construcción de la aplicación. Se analizaron los principios de diseño de la interfaz y los estándares de codificación a emplear durante la fase de construcción. Todo esto bien representado no solo en este documento sino además consultable por cualquier programador desde la misma ayuda de la aplicación gracias a la documentación generada por el Doxygen.

CONCLUSIONES

Una vez finalizado el desarrollo del presente trabajo se puede llegar a las siguientes conclusiones:

1. El estudio de las tendencias mundiales, herramientas y estándares requeridos para el desarrollo de software educativo en formato multimedia ha permitido determinar las funcionalidades del sistema utilizando tecnologías libres.
2. Se han logrado obtener los requerimientos del sistema mediante el uso de las técnicas de obtención de información descritas.
3. La correcta aplicación de la metodología RUP, el lenguaje UML y las herramientas seleccionadas han permitido la modelación del sistema para el desarrollo de software educativo en formato multimedia.
4. La alta motivación del equipo de desarrollo y la experiencia en el lenguaje de programación permitieron un rápido aprendizaje y dominio de las nuevas herramientas de trabajo utilizadas para la implementación del subsistema de edición visual.

En el presente trabajo no solo se les propuso, a la Dirección de Producción # 2 y a la Facultad 8 de la UCI, una solución a los problemas de migración al software libre, sino que además se dio origen a un producto base para el perfeccionamiento de la solución. Culminado el desarrollo del sistema se dio cumplimiento, de forma satisfactoria, a los objetivos que se perseguían inicialmente:

- Se fundamentó el alcance del proyecto comparado con las soluciones existentes y demostrando que es posible hacer software libre con los recursos y medios que se tienen en el país.
- Se demostró que existe una voluntad de trabajo dirigida al desarrollo de proyectos que beneficien la industria nacional en campos tradicionalmente desplazados por el mundo capitalista.
- Se logró crear una documentación amplia y fuerte en cuanto a las alternativas para el desarrollo de multimedia y software educativo siguiendo los principios del software libre.
- Se logró diseñar e implementar un subsistema que domina de forma eficiente las funcionalidades básicas para la edición de recursos multimedia dentro de un software educativo.

- Se logró diseñar e implementar un subsistema que permitirá al desarrollador obtener un proyecto exportable con extensión swf.
- Se logró definir una arquitectura sólida para el desarrollo de aplicaciones que a su vez es flexible a otras formas de producir multimedia educativa.
- Se obtuvo una herramienta que sigue los principios del software libre.

Se concluye que el sistema brinda una solución factible a la problemática inicial y que su desarrollo y explotación significará una mejora considerable en las facilidades de producción nacional e internacional de software educativo en formato multimedia para la UCI. El sistema además significa un paso de avance en la búsqueda de la independencia tecnológica de la sociedad. El mismo deja sentadas las bases para el inicio de las revoluciones que depara el futuro en estos campos.

RECOMENDACIONES

Toda obra humana es perfectible y las que constituyen un resultado intelectual están sujetas al constante cambio producto de la superación de sus creadores. Este trabajo no se encuentra exento de esto y se propone por tanto que se tomen en consideración aquellos elementos que por cuestiones de tiempo o recursos no fue posible incorporar. Se recomienda que futuros desarrollos de este producto consulten las siguientes sugerencias:

1. Portar la solución a otros sistemas operativos.
2. Modelar la solución considerando que la edición de elementos externos de tipo swf previamente creados pueda llevarse a cabo dentro del IDE.
3. Desarrollar en el swf exportado la instanciación de recursos por escena a partir del SWFMill.
4. Mejorar las técnicas de dibujado incluyendo nuevas herramientas y perfeccionando las posibilidades de configuración y funcionamiento de las actuales.
5. Reconstruir la interfaz de usuario aplicando los estándares de diseño gráfico definidos por la imagen de la UCI.
6. Incluir dentro de la línea de tiempo del IDE el trabajo con capas para un mejor aprovechamiento de la profundidad en la disposición espacial de los objetos en la escena.
7. Estudiar las posibilidades de incluir en el IDE la pre visualización del proyecto exportable o swf haciendo uso de un reproductor externo o interno.

También se recomienda darle un tiempo de prueba a este producto antes de desarrollar futuras versiones para estudiar la factibilidad de modificar algunas de las funcionalidades.

Entre las recomendaciones principales para la mejora del producto se encuentra el trabajo en el proceso de integración con otros dos subsistemas que se desarrollan en paralelo al mismo para la construcción del IDE. En vista a ello deben considerarse cuestiones como:

1- Una integración total con la librería de ejercicio, clases y componentes reutilizables que brindará al usuario del sistema la posibilidad de hacer uso de los componentes y objetos incluidos en la misma para enriquecer la interactividad del producto en edición.

2- Una integración más estrecha entre el editor de código y el sistema estará esencialmente basada en el intercambio de información a partir de los archivos de configuración de cada uno de los proyectos. El fichero de configuración del sistema contiene información relevante para que el editor de código pueda inyectar código directamente en el producto generado.

3- El uso de una interfaz de consola para el acceso al sistema en pos de generar productos a partir de archivos de configuración independientes será una solución práctica para que el editor de código mantenga actualizada sus interfaces.

REFERENCIAS BIBLIOGRÁFICAS

ARECHABALETA, M. G. 2008. Las plataformas de teleformación. [En línea] 2008. http://www.horizonteweb.com/magazine/comunet2.htm .

BÖHME, R. 2001. SWFTOOLS. [En línea] 2001. www.swftools.org/download.html.

CARREIRA, J. M. F. 2007. Formatos gráficos. [En línea] 2007. http://www.hsa.es/id/investigacion/uai/uai_docs/informatica/graficos.pdf.

CARRILLO, E. Z. 2008. LA EDUCACIÓN Y EL DESARROLLO GLOBAL MEDIADO POR LAS NUEVAS TECNOLOGÍAS DE LA INFORMACIÓN Y LA COMUNICACIÓN (TIC'S). [En línea] 2008. www.eaprender.org/edgarzamoracarrillo/laeducaci%C3%B3nyeldesarrolloglobalmediadoporlasnuevastecnolog%C3%ADdelainformaci%C3%B3nylacomunicaci%C3%B3ntic%E2%80%99s.

DÍAZ, R. E. B. 2007. Educación Virtual: Aulas sin Paredes. [En línea] 2007. www.educar.org/articulos/educacionvirtual.asp.

DIGIMATCH. 2007. Multimedia Educativa. [En línea] 2007. http://multimediaeducativa.zoomblog.com.

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. 1995. *Design Patterns. Elements of Reusable Object-Oriented Software.* 1995.

FOUNDATION, I. FREE SOFTWARE. 2007. GNU General Public License. [En línea] 2007. www.gnu.org/licenses/gpl.html.

LARA, R. G. 1997. Entrevista sobre QT Designer. [En línea] 1997. http://www.monografias.com/trabajos15/qt-designer/qt-designer.shtml.

LARMAN, C. 2003. *UML y Patrones.* 2003.

LILLEY, C. 2004. Scalable Vector Graphics (SVG). [En línea] 2004. www.w3.org/Graphics/SVG/.

OSFLASH. 2006. Motion-Twin ActionScript 2 Compiler. [En línea] 2006. www.mtasc.org/.

Pressman, R. S. 2002. *Ingeniería del software. Un enfoque práctico.* 2002.

STALLMAN. 2007. La Definición de Software Libre - Proyecto GNU - Fundación para el Software Libre (FSF). [En línea] 2007. http://www.gnu.org/philosophy/free-sw.es.html.

Trolltech. 2007. Mejoras en Qt 4.2. [En línea] 2007. http://doc.trolltech.com/4.2/qt4-2-intro.html.

URLICH, K. 2002. Macromedia Flash MX. [En línea] 2002. http://books.google.com/books?hl=es&lr=&id=AzNfZnOo6SAC&oi=fnd&pg=PR13&dq=que+es+macromedia+flash&ots=qrZJioQwD6&sig=uvQIZwDjzs7_yoLTSgTZDFtPnUA#PPR20,M1.

RAFFARIN, J.-P., & AILLAGON, J.-J. (2003, Noviembre 12). *N° 1206 - Projet de loi relatif au droit d'auteur et aux droits voisins dans la société de l'information.* Retrieved 2008, from Assemblée nationale ~ Les députés, le vote de la loi, le Parlement français: http://www.assemblee-nationale.fr/12/projets/pl1206.asp

BIBLIOGRAFÍA

CANTABRIA, G. D. 2004. Qué es GNU/Linux. [En línea] 2004. http://www.linuxglobal.org/linuxglobal/gnu.

COHEN, D. 2008. Zotero - Guía rápida. [En línea] 2008. www.zotero.org/documentation/guia_de_inicio_rapido.

ESMERALDASBLOG. 2007. Pencil, Software Gratuito para Diseño y Animación. [En línea] 2007. http://www.esmeraldasblog.com/2007/12/pencil-software-gratuito-para-diseo-y.html.

FISCHER, D. 2007. swfmill / swf2xml / xml2swf. [En línea] 2007. http://www.swfmill.org.

G.OSIPOV. 1988. *Libro de trabajo del sociólogo.* La Habana Cuba : Ciencias Sociales, 1988.

GEHRMANN, B. 1998. KDevelop - un Entorno de Desarrollo Integrado. [En línea] 1998. www.kdevelop.org/index.html?filename=main.html&set_lang=es.

GÓMEZ, M. A. 1998. Educación a Distancia. [En línea] 1998. http://www.sld.cu/libros/distancia/indice.html.

GONZÁLEZ, G. 2007. KToon: Herramienta de Animación en 2D. [En línea] 2007. http://ktoon-es.toonka.com/.

GONZÁLEZ, S. C. 2002. *Diseño teórico. EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA.* 2002.

HARRINGTON, B. 2003. EsFAQ - Inkscape. [En línea] 2003. //wiki.inkscape.org/wiki/index.php/EsFAQ.

HEESCH, D. V. 2005. KDevelop API Documentation. [En línea] 2005. http://www.kdevelop.org/HEAD/doc/api/html/classCppSupportPart.html.

LAFLECHA, Noticiario. 2007. Las declaraciones de Linus Torvalds y Richard Stallman en el tribunal. [En línea] 2007. http://www.laflecha.net/canales/curiosidades/200311156.

MATÍAS, S. P. 2007. Uira: flash en Linux. [En línea] 2007. http://www.etynos.org/web/2006/07/09/uira-flash-en-linux.

—. 2007. swfmill Open Source Flash. [En línea] 2007. http://osflash.org/swfmill.

Stallman. 2007. ¿Qué es Copyleft? [En línea] 2007. www.gnu.org/copyleft/copyleft.es.html.

STALLMAN, R. M. 2004. Software Libre para una sociedad libre. [En línea] 2004. http://biblioweb.sindominio.net/pensamiento/softlibre/softlibre.pdf.

WIKILIBROS. 2007. Programación con Qt4. [En línea] 2007. http://es.wikibooks.org/wiki/Programaci%C3%B3n:Qt4.

WAYNER, P. 2001. La ofensiva del software libre. [En línea] 2001. http://books.google.com/books?hl=es&lr=&id=eFuBwP6apJMC&oi=fnd&pg=PA9&dq=%22Wayner%22+%22La+ofensiva+del+software+libre::+C%C3%B3mo+Linux+y+el+â‹¯%22+&ots=5DTxtF7IhL&sig=AjJR4VIyr7WMsOAxjpBICfNhg54#PPA13,M1.

GLOSARIO DE TERMINOS

ADL

Los ADLs permiten modelar una arquitectura mucho antes que se lleve a cabo la programación de las aplicaciones que la componen, analizar su adecuación, determinar sus puntos críticos y eventualmente simular su comportamiento.

Análisis (Flujo de Trabajo)

Flujo de trabajo fundamental cuyo propósito principal es analizar los requisitos descritos en la captura de requisitos, mediante su refinamiento y estructuración. El objetivo de esto es: lograr una comprensión más precisa de los requisitos y obtener una descripción de los requisitos que sea más fácil de mantener y que ayude a dar estructura al sistema en su conjunto.

Aplicación (sistema)

Sistema que ofrece a un usuario final un conjunto coherente de casos de uso.

Arquitectura

Conjunto de decisiones significativas acerca de la organización de un sistema software, la selección de los elementos estructurales a partir de los cuales se compone el sistema, y las interfaces entre ellos, junto con su comportamiento, tal y como se especifica en las colaboraciones entre esos elementos, la composición de estos elementos estructurales y de comportamiento en subsistemas progresivamente mayores, y el estilo arquitectónico que guía está organización: estos elementos y sus interfaces, sus colaboraciones y su composición. La arquitectura del software se interesa no solo por la estructura y el comportamiento, sino también por las restricciones y compromisos de uso, funcionalidad, flexibilidad al cambio, funcionamiento, reutilización, comprensión, economía y tecnología, así como por aspectos estéticos.

Artefacto

Pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar actividades; representa un área de responsabilidad, y es candidata a ser tenida en cuenta para el control de la configuración. Un artefacto puede ser un modelo, un elemento de un modelo, o un documento.

CASE

Ingeniería de Software Asistida por Ordenador. Computer Aided Software Engineering

Caso de Uso

Es una descripción de un conjunto de secuencia de acciones, incluyendo variaciones, que un sistema lleva a cabo y que conduce a un resultado observable de interés para un actor determinado.

Doxygen

Es un programa de Dimitri van Heesch con licencia GPL que te permite la creación de documentación a partir del código fuente de un programa. Para esto, Doxygen utiliza la gramática del lenguaje en que esté escrito el código, así como los comentarios en un formato particular. Los lenguajes que soporta son: C, C++, Java, Objective C, Python, IDL y, en algunos casos, PHP, C# y D.

DTD

Siglas en inglés de Document Type Definition. La definición de tipo de documento (DTD) es una descripción de estructura y sintaxis de un documento XML o SGML. Su función básica es la descripción del formato de datos, para usar un formato común y mantener la consistencia entre todos los documentos que utilicen la misma DTD. De esta forma, dichos documentos, pueden ser validados, conocen la estructura de los elementos y la descripción de los datos que trae consigo cada documento, y pueden además compartir la misma descripción y forma de validación dentro de un grupo de trabajo que usa el mismo tipo de información.

Entidades

Representa un contenedor de información, algo físico que se utilice en el proceso del negocio y que sirva para obtener información o para actualizar información. Generalmente tiene estados, en dependencia de en qué momento aparezca como parte del proceso.

Fase

Período de tiempo entre dos hitos principales de un proceso de desarrollo.

Framework

Es una estructura de soporte definida, en la cual otro proyecto de software puede ser organizado y desarrollado. Puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

GUI

GUI (en inglés Graphical User Interface): Interfaz Gráfica de Usuario es un tipo de interfaz de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Habitualmente las acciones se realizan mediante manipulación directa para facilitar la interacción del usuario con la computadora.

IBM

International Business Machines o IBM (conocida coloquialmente como el Gigante Azul) es una empresa que fabrica y comercializa herramientas, programas y servicios relacionados con la informática. Tiene su sede en Armonk (Estados Unidos) y está constituida como tal desde el 15 de junio de 1911, pero lleva operando desde 1888.

Ingeniería de Software

Disciplina de la Ingeniería que concierne a todos los aspectos de la producción de software. Es una parte de la Ingeniería de la Ingeniería de Sistemas (concierne a todos los aspectos del desarrollo de sistemas basados en cómputo, que incluyen hardware, software y el proceso de Ingeniería).

IEEE

Instituto de IEEE de los ingenieros electrónicos eléctricos. Fundado en 1963, IEEE es una organización integrada por ingenieros, científicos, y estudiantes. IEEE es el mejor conocido para los estándares que se convierten para la industria del ordenador y del elemento electrónico.

IDE

Ambientes de Desarrollo Integrados. IDE (Integrated Development Environment): Un entorno de desarrollo integrado, es un programa compuesto por un conjunto de herramientas para un programador. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

LAME

LAME (acrónimo recursivo de LAME Ain't an MP3 Encoder) es un codificador de MPEG Audio Layer III (MP3) que puede ser usado con la mayoría de programas que convierten archivos WAV a archivos MP3 o desde otros formatos o soportes. Destaca por su espectacular rapidez, por la posibilidad de elegir la tasa de bits y el modo joint stereo. Este software se encuentra bajo una licencia de código abierto

Línea base

Conjunto de artefactos revisados y aprobados que representa un punto de acuerdo para la posterior evolución y desarrollo, y solamente puede ser modificado a través de un procedimiento formal, como la gestión de cambios y configuraciones.

Línea base de la arquitectura

Línea base resultado de la fase de elaboración.

Metodologías

Se encargan de elaborar estrategias de desarrollo de software que promuevan prácticas adoptativas en vez de predictivas; centradas en las personas o los equipos, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa del cliente.

Modelos

Es una descripción de (parte de) un sistema, descrito en un lenguaje bien definido. Un lenguaje bien definido es un lenguaje con una sintaxis y semántica precisa y que puede ser interpretado y manipulado por un ordenador.

MIC

Ministerio de la Informática y las Comunicaciones, es el organismo rector de estas disciplinas en Cuba.

Patrón

En la tecnología de objetos un patrón es una descripción de un problema y la solución, a la que se le da un nombre, y que se puede aplicar a nuevos contextos.

Renderización

La renderización es el proceso de generar una imagen desde un modelo. La palabra renderización proviene del inglés render. En términos de visualizaciones en ordenador, más específicamente en 3D, la "renderización" es un proceso de cálculo complejo desarrollado por un ordenador destinado a generar una imagen 2D a partir de una escena.

Requisitos

Condición o capacidad que debe cumplir un sistema. Requisitos Funcionales: especifica una acción que debe ser capaz de realizar el sistema. Requisito No Funcional: especifica restricciones físicas sobre un requisito funcional, tales como restricciones del entorno o de implementación, rendimiento, dependencias de la plataforma.

Responsabilidad

UML define una responsabilidad como “un contrato u obligación de un clasificador”. Las responsabilidades están relacionadas con las obligaciones de un objeto en cuanto a su comportamiento. Básicamente, estas responsabilidades son de los siguientes dos tipos:

Conocer: Conocer los datos privados encapsulados, los objetos relacionados y las cosas que puede derivar o calcular.

Hacer: Hacer algo él mismo, como crear un objeto o hacer un cálculo. Iniciar una acción en otros objetos. Controlar y coordinar actividades en otros objetos.

RSA

IBM Rational Software Architect (RSA) es una herramienta de diseño y desarrollo integrados que potencia el desarrollo orientado al modelado con UML para la creación de aplicaciones y servicios con buena arquitectura.

RUP

Proceso Unificado de Software (Rational Unified Process): Es un proceso de desarrollo de software. Forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo. Es una guía de cómo usar UML de la forma más efectiva.

Software

Es la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo.

UCI

Universidad de Ciencias Informáticas

UML

El Lenguaje Unificado de Modelado prescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

URL

Las siglas vienen de Uniform Resource Locator (en español: "Localizador Uniforme de Recursos"), se presenta como una secuencia de caracteres bajo una forma estándar para darle nombre a determinados recursos en una red. Entre los diferentes esquemas de URLs se podría mencionar: Http (hipertexto), https (hipertexto seguro), ftp (protocolo de transferencia de archivos), mailto (correo electrónico), file (archivos locales).

XML

Extensible Markup Language (lenguaje de Marcas Extensibles). Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Permite definir la gramática de lenguajes específicos de la misma manera que HTML.

XP

eXtreme Programming. Es una metodología de desarrollo de software.

Zlib

Zlib es una biblioteca de compresión de datos, de software libre/fuente abierta, multiplataforma desarrollada por Jean-loup Gailly y Mark Adler. Esta biblioteca provee una implementación del algoritmo DEFLATE usado en el programa de compresión gzip. La primera versión pública, 0.9, fue lanzada el 1 de mayo de 1995 y fue originalmente orientada para ser usada con la librería de imágenes libpng. La biblioteca zlib es distribuida bajo la licencia zlib.

ANEXOS

ANEXO A “Descripción de los Casos de Uso del Sistema”

Caso de Uso:	Gestionar Proyecto
Actores:	Desarrollador
Resumen:	El caso de uso procede cuando el desarrollador decide crear un nuevo proyecto, exportar, realizar salvas o cerrar el proyecto en que está trabajando.
Referencia:	R1,R2, R3, R4, R5
CU asociados:	
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.0 El desarrollador accederá al menú Archivo. (1)	1.1 El sistema desplegará las siguientes opciones del menú: a. Nuevo Proyecto (2) b. Abrir Proyecto (3) c. Guardar Proyecto (4) d. Guardar Como (5) e. Exportar Proyecto (6) f. Cerrar Proyecto g. Salir. (7)
2.0 El desarrollador seleccionará una de las opciones.	2.1 El sistema conducirá a una de las secciones de acuerdo a la opción seleccionada: a. Nuevo Proyecto(Sección “Nuevo Proyecto”) b. Abrir Proyecto (Sección “Abrir Proyecto”) c. Guardar Proyecto (Sección “Guardar Proyecto”) d. Guardar Como (Sección “Guardar Como Proyecto”)

	e. Exportar Proyecto (Sección “Exportar Proyecto”) f. Cerrar Proyecto (Sección “Cerrar Proyecto”) g. Salir(Sección “Salir”)
Sección “Nuevo Proyecto”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema verifica si hay un proyecto abierto y en tal caso llama a la sección “Cerrar Proyecto”.
	2.0 El sistema activa las ventanas del entorno de trabajo ubicándolas a un estado por defecto predeterminado anteriormente: Un fotograma clave activo en la Línea de Tiempo, El escenario vacío, la biblioteca vacía, activa la Herramienta puntero del Panel de Herramientas, y el panel de Dibujado con una configuración por defecto.
Sección ”Abrir Proyecto”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema verifica la existencia de un proyecto abierto llamando a la sección “Cerrar Proyecto”. 1.1 El sistema mostrará una ventana de dialogo para realizar la búsqueda del proyecto que se desee abrir.
2.0 El desarrollador localizará en la ventana de diálogo mostrada la ubicación del archivo de proyecto que desea abrir. (8)	2.1 El sistema cargará las distintas opciones que se encuentren presentes en el archivo de proyecto.
	3.1 El sistema cargará las interfaces principales del IDE. 3.2 El sistema cargará y visualizará en la biblioteca todos los elementos especificados en el proyecto con anterioridad.

	3.2 El sistema cargará y visualizará en la Línea de Tiempo la secuencia de fotogramas correspondientes. 3.3 El sistema carga en el Escenario la escena correspondiente al primer fotograma clave de la Línea de Tiempo.
Sección “Guardar Proyecto”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema comprueba si existe un fichero de proyecto, de no existir llama a la sección “Guardar Como Proyecto”.
	2.0 El sistema toma los cambios registrados en el entorno de trabajo y modifica el fichero de proyecto. 2.1 El sistema llamará al CUS Gestionar Cambios.
Sección “Guardar Como Proyecto”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema mostrará una ventana de diálogo para la búsqueda del nuevo directorio donde el desarrollador desee salvar la nueva copia y brindará la posibilidad del cambio de nombre del proyecto.
2.0 El desarrollador seleccionará el nuevo directorio y el nuevo nombre del proyecto. (9)	2.1 El sistema hace una copia del fichero anterior hacia la nueva dirección y con el nombre introducido por el desarrollador.
	3.0 El sistema llama a la sección “Guardar Proyecto”.
Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
	2.1.a El sistema crea un nuevo fichero vacío con los datos especificados por el desarrollador.

Sección “Exportar Proyecto”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema llamará a la sección “Guardar Proyecto”.
	2.0 El sistema cargará del archivo de proyecto la información necesaria para generar de él dos nuevos archivos temporales en el directorio de trabajo del proyecto: 1- El primero, un xml con la información relativa a la biblioteca que interpretará el SWFmill 2- El segundo, un archivo de texto con la información relativa al escenario que interpretará el SWFC
	3.0 El sistema hará un llamado al sistema SWFmill para que genere un primer archivo swf como biblioteca base del producto final. 3.1 El sistema hará un llamado al sistema SWFC para que genere el archivo swf final del producto a partir del fichero que le corresponde y la biblioteca anteriormente generada como swf.
Sección “Cerrar Proyecto”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema comprobará si el proyecto no se ha modificado desde la última salva. 1.1 El sistema mostrará una ventana de dialogo brindando la opción de salvar los nuevos cambios o deshacerlos.
2.0 El desarrollador aceptará salvar los nuevos cambios. (9)	2.1 El sistema conducirá a la Sección “Guardar Proyecto”.

	3.0 El sistema desactivará todas las funcionalidades del IDE dejando solo las necesarias para consultar ayudas y para la creación o la apertura de nuevos proyectos.
Sección “Salir”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema revisará si existe un proyecto abierto, conduciendo a la Sección “Cerrar”. 1.1 El sistema cerrará la ventana principal.
Poscondiciones:	
Prioridad:	Critico
Especificaciones Complementaria:	
Prototipos de Interfaz	

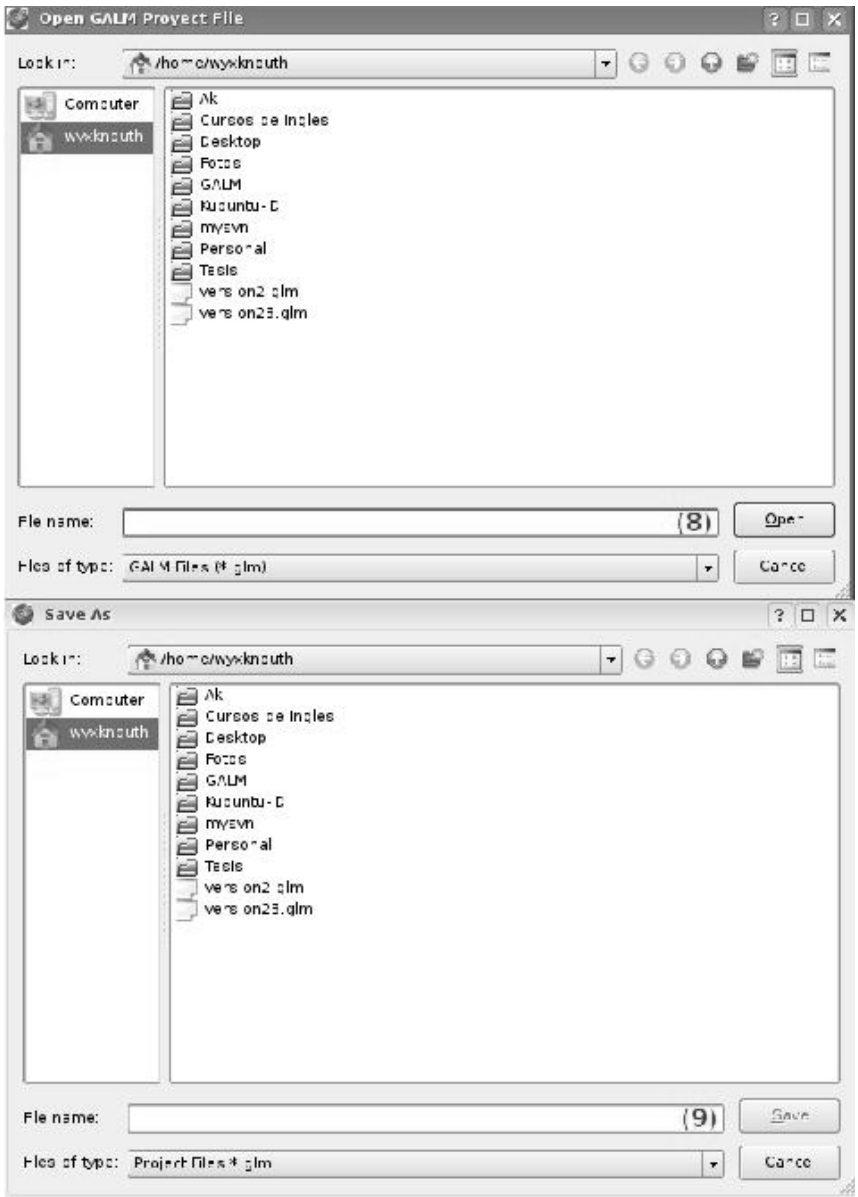


Tabla A.1 Descripción del CUS 1

Caso de Uso:	Gestionar Opciones
Actores:	Desarrollador
Resumen:	El caso de Uso procede cuando se decide configurar las opciones del proyecto relativas al escenario y al fichero de salida swf.
Referencia:	R6, R7, R8, R9, R10, R11
CU asociados:	
Precondiciones:	

<table>
<tr><th colspan="3">Flujo Normal de Eventos</th></tr>
<tr><th>Acción del Actor</th><th colspan="2">Respuesta del Sistema</th></tr>
<tr><td>1. El desarrollador accederá al menú Archivo --> opciones del proyecto. (1)</td><td colspan="2">1.0 El sistema mostrará una ventana con dos pestañas donde en la primera se mostrarán las opciones de exportación (2) del proyecto tales como nombre y versión del reproductor con que se generará la salida y en la otra las opciones de la película (3) como velocidad de frames por segundos, ancho y alto de la escena.</td></tr>
<tr><td>2. El desarrollador modificará las opciones que crea necesario y necesite.</td><td colspan="2">2.1. El sistema guardará esas opciones en el fichero del proyecto.
2.2 El sistema aplicará al escenario las opciones que fueron modificadas.</td></tr>
<tr><td>Poscondiciones:</td><td colspan="2"></td></tr>
<tr><td>Prioridad:</td><td colspan="2">Critico</td></tr>
<tr><td>Especificaciones Complementaria:</td><td colspan="2"></td></tr>
<tr><td>Prototipos de Interfaz</td><td colspan="2"> 
</td></tr>
</table>

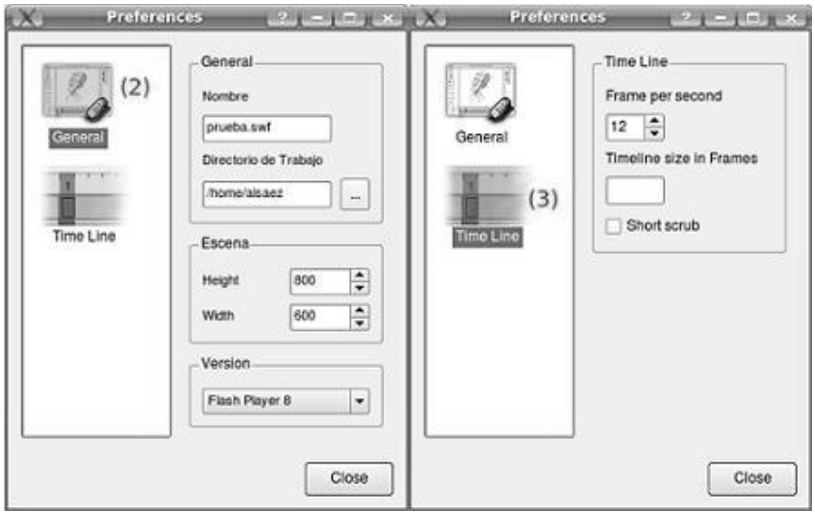
	

Tabla A.2 Descripción del CUS2

Caso de Uso:	Gestionar Herramientas
Actores:	Desarrollador
Resumen:	El caso de uso procede cuando el desarrollador decide activar, desactivar o configurar a través del menú visual alguna de las herramientas disponibles para el trabajo con los recursos del proyecto dentro del área de trabajo o escenario.
Referencia:	R43, R44, R45, R46, R47, R48, R49, R50
CU asociados:	
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1- El desarrollador seleccionará una de las herramientas del panel de herramientas. (1)	1.1- El sistema activará la herramienta seleccionada conduciéndola a la sección correspondiente 1. Herramienta Línea (Sección Activar Trazo) (2) 2. Herramienta Rectángulo (Sección Activar Forma) (3) 3. Herramienta Polilínea (Sección Activar Trazo) (4) 4. Herramienta Elipse (Sección Activar

	Forma) (5) 5. Herramienta Lápiz (Sección Activar Trazo) (6) 6. Herramienta Brocha (Sección Activar Trazo) 7. Herramienta Goma (Sección Activar Goma) 8. Herramienta Texto (Sección Activar Texto) (7) 9. Herramienta Relleno (Sección Activar Relleno) 10. Herramienta Selección (8) 11. Herramienta Zoom (9)
	2.0 El sistema cambiará el icono del mouse en dependencia de la herramienta activada. 2.1 El sistema sobresaltará el icono correspondiente a la herramienta activa.
Sección “Activar Trazo”	
Acción del Actor	**Respuesta del Sistema**
	1.1. El sistema mostrará en el panel de dibujado el tipo de trazo por defecto. (A) 1.2 El sistema mostrará en el panel de dibujado la opción de grosor por defecto.(B) 1.3 El sistema mostrará en el panel de dibujado el nombre y el número del color del trazo por defecto. (C)
Sección “Activar Forma”	
Acción del Actor	**Respuesta del Sistema**
	1.1. El sistema mostrará en el panel de dibujado el tipo de borde por defecto.(D) 1.2 El sistema mostrará en el panel de dibujado la opción de grosor del borde por defecto.(E) 1.3 El sistema mostrará en el panel de dibujado el nombre y el número del color del borde por defecto.(G) 1.4 El sistema mostrará en el panel de dibujado

	el nombre y el número del color del relleno por defecto. (F)
Sección “Activar Goma”	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema mostrará en el panel de dibujado la opción del tamaño de la goma por defecto.
Sección “Activar Texto”	
Acción del Actor	**Respuesta del Sistema**
	1.1. El sistema mostrará en el panel de dibujado el tipo de fuente por defecto. (H) 1.2 El sistema mostrará en el panel de dibujado el tamaño de fuente por defecto.(I) 1.3 El sistema mostrará en el panel de (J) dibujado el color de la fuente por defecto.
Sección “Activar Relleno”	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema mostrará en el panel de dibujado el color de relleno por defecto. 1.2 El sistema mostrará en el panel de dibujado el gradiente de relleno por defecto. 1.3 El sistema mostrará en el panel de dibujado la textura de relleno por defecto.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.0 El desarrollador modificará alguna de las opciones del panel de dibujado. (10)	1.1. El sistema modificará la propiedad correspondiente de la herramienta activa.

Poscondiciones:	
Prioridad:	Crítico
Especificaciones Complementaria:	
Prototipo de Interfaz	

Tabla A.3 Descripción del CUS 3

Caso de Uso:	CU Gestionar Biblioteca
Actores:	Desarrollador
Resumen:	El CU procede cuando el desarrollador desea realizar alguna modificación sobre la biblioteca o los elementos almacenados en ella.
Referencia:	R12, R13, R14, R15
CU asociados:	CUS Gestionar Línea de Tiempo, CUS Gestionar Inspector de Propiedades
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.0 El desarrollador seleccionará un recurso dentro de la biblioteca.	1.1El sistema activará las siguientes opciones: a. Exportar recurso al escenario b. Eliminar recurso
2.0 El desarrollador seleccionará una de las opciones posibles. (2)(3)	2.1 El sistema llamará a la sección correspondiente de acuerdo a la selección del desarrollador:

	a. Exportar recurso al escenario (CUS Gestionar Línea de Tiempo) b. Eliminar recurso (Sección Eliminar Recurso) (2)
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.0.a El desarrollador seleccionará la opción de importar un nuevo recurso a la biblioteca.(1)	1.1.a El sistema abrirá una ventana de búsqueda mostrando el sistema de almacenamiento de la computadora.
2.0.a El desarrollador recorrerá la jerarquía de carpetas hasta encontrar el fichero.	2.1.a El sistema mostrará los ficheros que es capaz de importar en cada jerarquía de carpetas que se visita.
3.0 El desarrollador seleccionará el fichero y presionará el botón open. (1.1)	3.1 El sistema adjuntará al proyecto el recurso seleccionado. 3.2 El sistema representará el recurso importado en la biblioteca de recursos.
	4.0 El sistema llamará al CU Gestionar Cambios.
Sección “Eliminar Recurso”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema eliminará el recurso de la lista de recursos de la biblioteca. 1.1 El sistema llamará al CUS Gestionar Línea de Tiempo.
	2.0 El sistema llamará al CU Gestionar Cambios.

Poscondiciones:	
Prioridad:	Crítico
Especificaciones Complementaria:	

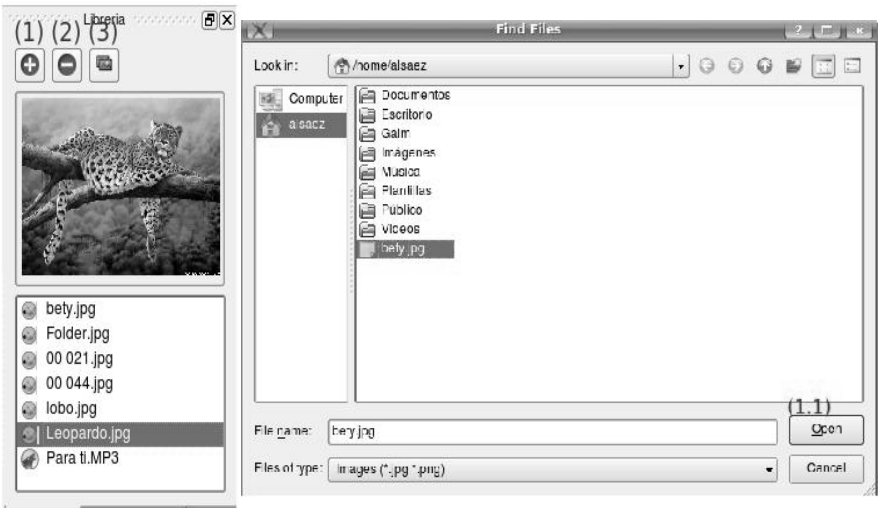
Interfaz	

Tabla A.4 Descripción del CUS 8

Caso de Uso:	CUS Gestionar Línea de Tiempo
Actores:	Desarrollador
Resumen:	El CU procede cuando el desarrollador va a realizar alguna acción relacionada con la línea de tiempo como creación de fotogramas, fotogramas activos, así como acciones que modifiquen permanentemente la escena del fotograma clave activo.
Referencia:	R18, R19, R20, R21, R22, R23, R24, R25
CU asociados:	CUS Gestionar Escenario
Precondiciones:	Debe existir desde la creación del proyecto un fotograma clave por defecto en la línea de tiempo.

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El desarrollador seleccionará uno de los fotogramas de la línea de tiempo. (1)	1.1 El sistema resaltará el fotograma activo dentro de la línea de tiempo. 1.2 El sistema llamará al CUS Gestionar Escenario para el fotograma clave activo en ese momento.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**

1.0 El desarrollador dará clic derecho sobre un fotograma de la línea de tiempo.	1.1 El sistema mostrará un listado de acciones que pueden estar activas o no en dependencia del tipo de fotograma seleccionado. a. Eliminar Fotograma (3) b. Eliminar Fotograma Clave. (3) c. Crear Fotograma. (2) d. Crear Fotograma Clave. (4) e. Transformar Fotograma a Fotograma Clave(4)
2.0 El desarrollador selecciona una de las opciones mostradas	2.1 En caso de seleccionar la opción: A, va a la sección Eliminar Fotograma. B, va a la sección Eliminar Fotograma. C, va a la sección Crear Fotograma. D, va a la sección Crear Fotograma Clave. E, va a la sección Convertir Fotograma en Fotograma Clave.
Sección “Eliminar Fotograma”	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema determinará a que fotograma clave pertenecía el fotograma. 1.2 El sistema reducirá la duración de la escena correspondiente al fotograma clave encontrado. 1.3 El sistema eliminará la representación visual del fotograma en la línea de tiempo.
	2.0 El sistema llamará al CU Gestionar Cambios.
Sección “Eliminar Fotograma Clave”	
Acción del Actor	**Respuesta del Sistema**
	1.1El sistema ratificará que es posible continuar reduciendo la duración de su escena. 1.2 El sistema llamará a la sección Eliminar Fotograma para el fotograma representado visualmente detrás de él en la línea de tiempo.

	2.0 El sistema llamará al CU Gestionar Cambios.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.1.a El sistema eliminará el fotograma clave seleccionado de la lista de la línea de tiempo. 1.2 El sistema eliminará la representación visual de dicho fotograma clave en la línea de tiempo. 1.3 El sistema llamará al CUS Gestionar Escenario para el fotograma clave activo en ese momento.
Sección “Crear Fotograma”	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema representará visualmente el fotograma. 1.2 El sistema incrementará la duración de la escena a la que representa el fotograma.
	2.0 El sistema llamará al CU Gestionar Cambios.
Sección “Crear Fotograma Clave”	
Acción del Actor	**Respuesta del Sistema**
	1.1El sistema representará visualmente el fotograma clave en la línea de tiempo. 1.2.El sistema creará un fotograma clave vacío en la lista de la línea de tiempo. 1.3 El sistema asignará a la escena correspondiente tanta duración como fotogramas le precedan al fotograma clave creado. 1.4 El sistema reducirá la duración de la escena perteneciente al fotograma clave anterior al creado de acuerdo la duración de la escena del nuevo fotograma clave.

	1.5 El sistema llamará al CUS Gestionar Escenario para el fotograma clave activo en ese momento.
	2.0 El sistema llamará al CU Gestionar Cambios.
Sección “Convertir Fotograma en Fotograma Clave”	
Acción del Actor	**Respuesta del Sistema**
	1.1El sistema representará visualmente el fotograma clave en la línea de tiempo. 1.2El sistema creará un fotograma clave en la lista de la línea de tiempo. 1.3 El fotograma convertido a fotograma clave mantendrá la escena correspondiente al fotograma clave que le preceda. 1.4 El sistema asignará a la escena del nuevo fotograma clave tanta duración como fotogramas le precedan al fotograma clave creado. 1.5 El sistema reducirá la duración de la escena perteneciente al fotograma clave anterior al fotograma creado de acuerdo la duración de la escena del nuevo fotograma clave. 1.6 El sistema llamará al CUS Gestionar Escenario para el fotograma clave activo en ese momento.
	2.0 El sistema llamará al CU Gestionar Cambios.
Sección “Instanciar Recurso”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema creará una nueva instancia con la referencia del recurso que se encuentra en la biblioteca con las coordenadas que el escenario les provea. 1.1El sistema llamará al CUS Gestionar Escenario para el fotograma clave activo en ese

	momento.
	2.0 El sistema llamará al CU Gestionar Cambios.
Sección “Eliminar Instancias”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema recorre el listado de todos los fotogramas y elimina de ellos todas las instancias que hagan referencia al identificador que se desea eliminar. 1.1 El sistema llamará al CUS Gestionar Escenario para el fotograma clave activo en ese momento.
	2.0 El sistema llamará al CU Gestionar Cambios.
Sección “Eliminar una Instancia”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema elimina del fotograma especificado la instancia que hagan referencia al identificador de la misma. 1.1 El sistema llamará al CUS Gestionar Escenario para el fotograma clave activo en ese momento.
	2.0 El sistema llamará al CU Gestionar Cambios.
Sección “Adicionar Objeto Interno”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema guardará en la lista de objetos del fotograma clave activo el objeto creado por el escenario.
	2.0 El sistema llamará al CU Gestionar Cambios.
Sección “Eliminar Objeto Interno”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema buscará en la lista de objetos del fotograma clave activo el objeto seleccionado por el escenario.

	1.1 El sistema eliminará el objeto de la lista de objetos. 1.2 El sistema llamará al CUS Gestionar Escenario para el fotograma clave activo en ese momento.
	2.0 El sistema llamará al CU Gestionar Cambios.

Poscondiciones:	
Prioridad:	Secundario
Especificaciones Complementaria:	
Prototipos de Interfaz	

Tabla A.5 Descripción del CUS 4

Caso de Uso:	Gestionar Escenario
Actores:	Desarrollador
Resumen:	El caso de uso procede cuando el desarrollador decide trabajar en el escenario con la herramienta activa en el panel de herramientas.
Referencia:	R31, R32, R33, R34, R35, R36, R37, R38, R39, R40, R41, R42
CU asociados:	CUS Gestionar Herramientas, CUS Gestionar Inspector de Propiedades
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.0 El desarrollador dará clic izquierdo sobre el escenario.(1)	1.1 El sistema tomará del panel de herramientas la herramienta activa y sus propiedades 1.2 De acuerdo a la herramienta el escenario desarrollará el siguiente listado de actividades: 1. Herramienta Línea (Sección Dibujar Forma) 2. Herramienta Rectángulo (Sección Dibujar Forma)

	3. Herramienta Polilínea (Sección Dibujar Polilíneas) 4. Herramienta Elipse (Sección Dibujar Forma) 5. Herramienta Lápiz (Sección Dibujar Trazo) 6. Herramienta Texto (Sección Representar Texto) 7. Herramienta Selección (Sección Selección) 8. Herramienta Zoom (Sección Zoom)
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1. 0.a El desarrollador dará clic derecho sobre el escenario.	1.1.a El sistema verificará si el punto seleccionado pertenece a un área previamente enmarcada. 1.2.a El sistema llamará a la Sección Selección si el punto no pertenece a un área previamente enmarcada. 1.3.a El sistema habilitará las opciones de edición de escenario permitidas de acuerdo al área seleccionada. 1.4.a El sistema mostrará las opciones de edición habilitadas.
Sección “Dibujar Trazo”	
Acción del Actor	**Respuesta del Sistema**
1.0 El desarrollador arrastra el mouse.	1.1 El sistema adicionará puntos al objeto de referencia para trazos de acuerdo a la herramienta con que se trabaja en el escenario y sus propiedades, tomando los datos necesarios a partir de las acciones del desarrollador (ptos que componen el trazo...). 1.2 El sistema representará visualmente el objeto modificado.
2.0 El desarrollador liberará el mouse.	2.1 El sistema llamará al CUS Gestionar Línea

	de Tiempo. 2.2 El sistema representará visualmente el objeto creado. 2.3 El sistema eliminará el objeto de referencia para trazos.
	3.0 El sistema llamará a la Sección Selección
Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
	1.1.a El sistema creará un objeto de referencia para trazos de acuerdo a la herramienta con que se trabaja en el escenario y sus propiedades, tomando los datos necesarios a partir de las acciones del desarrollador (Punto inicial.). 1.2.a El sistema llamará a la sección “Dibujar Trazo”
Sección “Dibujar Forma”	
Acción del Actor	**Respuesta del Sistema**
1.0 El desarrollador arrastra el mouse.	1.1El sistema eliminará cualquier objeto temporal previamente visualizado en el escenario y deshará su visualización. 1.2 El sistema creará un objeto temporal de acuerdo a la herramienta con que se trabaja en el escenario y sus propiedades, tomando los datos necesarios a partir de las acciones del desarrollador (ptos de inicio y fin de la forma...). 1.3 El sistema representará visualmente el objeto creado temporalmente.
2.0 El desarrollador liberará el mouse.	2.1El sistema eliminará cualquier objeto temporal previamente visualizado en el escenario y deshará su visualización. 2.2 El sistema creará un objeto de acuerdo a la

	herramienta con que se trabaja en el escenario y sus propiedades, tomando los datos necesarios a partir de las acciones del desarrollador (ptos de inicio y fin de la forma...). 2.3 El sistema llamará al del CUS Gestionar Línea de Tiempo. 2.4 El sistema representará visualmente el objeto creado.
	3.0 El sistema llamará a la Sección Selección
Sección “Representar Texto”	
Acción del Actor	Respuesta del Sistema
	1.0 El sistema creará un objeto texto con un contenido por defecto. 1.1 El sistema llamará al CUS Gestionar Línea de Tiempo. 1.2 El sistema representará visualmente el objeto creado.
	2.0 El sistema llamará a la Sección Selección
Sección “Cargar Escena”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema eliminará toda representación visual que exista en el escenario. 1.1 El sistema visualizará en el escenario todas las instancias y objetos internos de la escena correspondiente al fotograma clave activo.
Sección “Dibujar Polilínea”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema consultará en el objeto de referencia para polilíneas los datos necesarios para continuar con su trazado.
2.0 El desarrollador arrastra el mouse.	2.1 El sistema modificará el objeto temporal de

	referencia para la polilínea. 2.2 El sistema actualizará la representación visual del objeto creado temporalmente.
3.0 El desarrollador liberará el mouse.	3.1 El sistema modificará el objeto temporal de referencia para la polilínea. 3.2 El sistema actualizará la representación visual del objeto creado temporalmente. 3.3 El sistema llamará al CUS Gestionar Inspector de Propiedades.
	4.0 El sistema llamará a la Sección Selección
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.0.a El sistema creará un objeto de acuerdo a la herramienta con que se trabaja en el escenario y sus propiedades, tomando los datos necesarios a partir de las acciones del desarrollador. 1.1.a El sistema creará un objeto temporal de referencia para polilíneas semejante al objeto creado anteriormente. 1.2.a El sistema llamará CUS Gestionar Línea de Tiempo.
Sección “Zoom”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema incrementará las dimensiones del escenario y las imágenes contenidas en el mismo. 1.1 El sistema modificará el icono de la herramienta en su panel indicando su nueva funcionalidad.(Zoom inverso).
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.0.a El sistema decrementará las dimensiones

	del escenario y las imágenes contenidas en el mismo. 1.1.a El sistema modificará el icono de la herramienta en su panel indicando su nueva funcionalidad.(Zoom).
Sección “Selección”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema eliminará cualquier delimitación de área previamente establecida. 1.1 El sistema determinará a que objeto pertenece el pto seleccionado en el escenario del fotograma clave activo. 1.2 El sistema delimitará el área que contiene completamente a ese objeto.
2.0 El desarrollador liberará el mouse.	2.1 El sistema llamará al CU Gestionar Inspector de Propiedades. 2.2 El sistema habilitará las opciones de edición de escenario permitidas de acuerdo al área seleccionada.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.0.a El desarrollador arrastrará el mouse sobre el escenario, enmarcando un área determinada y liberará el mouse.	1.1.a El sistema eliminará cualquier delimitación de área previamente establecida. 1.2.b El sistema determinará cuales objetos de la escena del fotograma clave activo están contenidos en el área. 1.3.a El sistema delimitará el área que contiene completamente a esos objetos. 1.4.a El sistema habilitará las opciones de edición de escenario permitidas de acuerdo al área seleccionada.

2.0.b El desarrollador arrastrará el mouse sosteniendo el clic.	2.1.b El sistema eliminará cualquier copia temporal del objeto previamente visualizada en el escenario y deshará su visualización. 2.2.b El sistema creará una copia temporal del recurso seleccionado a partir de sus propiedades pero modificando las coordenadas por las indicadas por el desarrollador durante el arrastre. 2.3.b El sistema representará visualmente la copia creado temporalmente. 2.4.b El sistema llamará al CU Gestionar Inspector de Propiedades para la copia temporal.
3.0.b El desarrollador liberará el mouse.	3.1.b El sistema llamará al CU Gestionar Inspector de Propiedades para el recurso inicialmente seleccionado modificando sus coordenadas con las coordenadas de la copia temporal. 3.2.b El sistema eliminará cualquier copia temporal del objeto previamente visualizada en el escenario y deshará su visualización. 3.3.b El sistema habilitará las opciones de edición de escenario permitidas según el recurso seleccionado.
Poscondiciones:	
Prioridad:	Crítico
Especificaciones Complementaria	

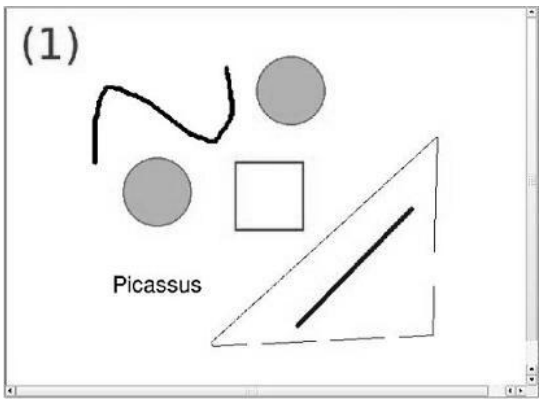
Prototipos de Interfaz	

Tabla A.6 Descripción del CUS 6

Caso de Uso:	CU Gestionar Inspector de Propiedades	
Actores:	Desarrollador	
Resumen:	El CU se procede a ejecutar cuando el desarrollador selecciona un elemento del proyecto, bien halla sido seleccionado en el escenario o en la biblioteca de recursos.	
Referencia:	R16,R17	
CU asociados:		
Precondiciones:		
Flujo Normal de Eventos		
Acción del Actor		**Respuesta del Sistema**
1.0. El desarrollador modificará algunas de las propiedades del recurso en el escenario o en los campos del inspector de propiedades.		1.1 El sistema modificará las propiedades del recurso mostrado. 1.2 El sistema llamará al CUS Gestionar Escenario para el fotograma clave activo en ese momento.
		2.0 El sistema llamará al CU Gestionar Cambios.
Sección “Mostrar Propiedades”		
Acción del Actor		**Respuesta del Sistema**
		1.1 El sistema mostrará una ventana en dependencia del tipo de objeto seleccionado y sus propiedades serán los campos mostrados para ser editados o mostrados.
Poscondiciones:		

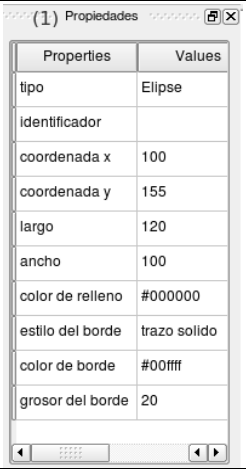
Prioridad:	Crítico
Especificaciones Complementaria:	
Prototipos de Interfaz	

Tabla A.7 Descripción del CUS 5

Caso de Uso:	Editar Escenario
Actores:	Desarrollador
Resumen:	El caso de uso procede cuando el desarrollador trabaja sobre el escenario con la herramienta selección luego de activar alguna de las opciones de edición que se muestran al dar clic derecho el desarrollador sobre algún recurso.
Referencia:	R26, R27, R28, R29, R30
CU asociados:	CUS Gestionar Escenario
Precondiciones:	

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El desarrollador seleccionará una de las opciones de edición mostradas por el sistema.	1.1 El sistema llamará a la sección correspondiente de acuerdo a la opción seleccionada: A. Eliminar (Sección Eliminar Recurso)(4) B. Copiar (Sección Copiar Recurso) (2) C. Cortar (Sección Cortar Recurso) (1) D. Pegar (Sección Pegar Recurso) (3)

Sección “Eliminar Recurso”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema llamará al CUS Línea de Tiempo.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	1.0.a El sistema llamará al CUS Línea de Tiempo.
Sección “Copiar Recurso”	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema eliminará cualquier copia temporal previamente creada. 1.2 El sistema creará una copia temporal del recurso seleccionado a partir de sus propiedades.
Sección “Cortar Recurso”	
Acción del Actor	**Respuesta del Sistema**
	1.1 El sistema eliminará cualquier copia temporal previamente creada. 1.2 El sistema creará una copia temporal del recurso seleccionado a partir de sus propiedades. 1.3 El sistema llamará a la sección Eliminar Recurso.
Sección “Pegar Recurso”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema llamará al CUS Línea de Tiempo para la copia temporal. 1.1 El sistema representará en el escenario la copia temporal previamente guardada. 1.2 El sistema llamará al CUS Gestionar Escenario.
Flujos Alternos	

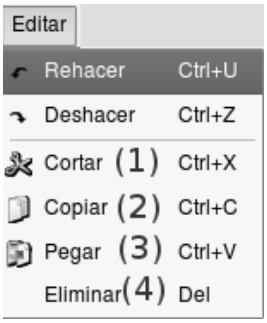
Acción del Actor	Respuesta del Sistema
	1.0.a El sistema llamará al CUS Línea de Tiempo para la copia temporal. 1.1.a El sistema llamará al CUS Gestionar Escenario.
Poscondiciones:	
Prioridad:	Secundario
Especificaciones Complementaria:	Los puntos extremos permiten incrementar o decrementar las dimensiones de un elemento interno así como girarlo en base a su centro. El centro del elemento interno también cuenta como uno de sus puntos extremos y puede ser modificado.
Prototipos de Interfaz	

Tabla A.8 Descripción del CUS 7

Caso de Uso:	CU Gestionar Cambios
Actores:	Desarrollador
Resumen:	El CU procede cuando se realiza alguna acción sobre el proyecto y se va guardando una traza del mismo de como estaba antes del cambio para poder deshacer o rehacer acciones.
Referencia:	R51,R52
CU asociados:	
Precondiciones:	
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1.0 El actor accederá al menú editar -> deshacer o a su acceso directo CTRL+Z (2)	1.1 El sistema accederá a su registro de acciones y recrea el estatus del proyecto un paso atrás en el historial del mismo dejando

	inactivo el cambio correspondiente.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
1.0.a El actor accederá al menú editar -> rehacer o a su acceso directo CTRL+U (1)	1.1 El sistema accederá a su registro de acciones y recrea el estatus del proyecto un paso adelante en el historial del mismo, activando el cambio correspondiente.
Sección “Adicionar Cambio”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema crear un objeto cambio con referencias a las acciones provenientes del escenario, biblioteca o inspector de propiedades. 1.1 El sistema determinará de acuerdo al tipo de cambio su necesidad de almacenar total o parcialmente los datos sin modificar. 1.2 El sistema almacenará en la lista de cambios el nuevo cambio.
Sección “Eliminar Historial”	
Acción del Actor	**Respuesta del Sistema**
	1.0 El sistema eliminará todos los cambios registrados con anterioridad.
Poscondiciones:	
Prioridad:	Opcional
Especificaciones Complementaria:	
Prototipos de Interfaz	

Tabla A.9 Tabla A.8 Descripción del CUS 7

ANEXO B “Prototipo de Interfaz”

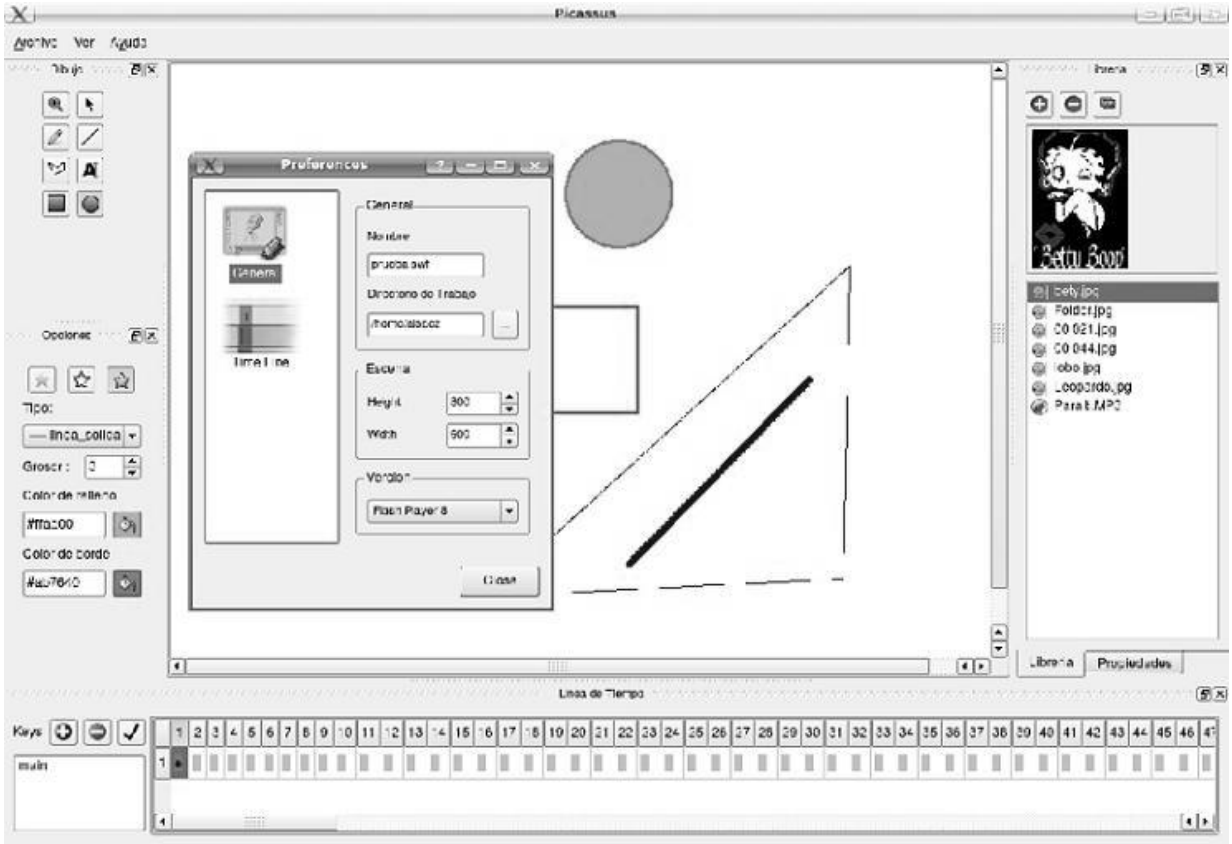


Figura B.1

ANEXO C “Estudio de Factibilidad”

Tipo	Descripción	Peso	Cant * peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface)	1	0*1
Medio	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto.	2	0*2
Complejo	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3	1*3
Total			3

Tabla C.1 Calcular UAW

Tipo	Descripción	Peso	Cant * peso
Simple	El Caso de Uso contiene de 1 a 3 transacciones	5	3*5
Medio	El Caso de Uso contiene de 4 a 7 transacciones	10	2*10
Complejo	El Caso de Uso contiene más de 8 transacciones	15	4*15
Total			95

Tabla C.2 Calcular UUCW

Factor	Descripción	Peso	Valor	Comentario	Σ ($Peso_i$ * $Valor_i$)
T1	Sistema distribuido	2	0	El sistema es centralizado	0
T2	Objetivos de	1	4	Se trabaja con	4

	rendimiento o tiempo de respuesta			gráficos	
T3	Eficiencia del usuario final	1	1	Escasas restricciones de eficiencia	1
T4	Procesamiento interno complejo	1	3	No son demasiado complejos los cálculos	3
T5	El código debe ser reutilizable	1	4	Se piensa en futuras versiones.	4
T6	Facilidad de instalación	0.5	1	Apenas es una versión de prueba.	0.5
T7	Facilidad de uso	0.5	3	En esta versión se prioriza la funcionalidad	1.5
T8	Portabilidad	2	1	Se debe facilitar para próximas versiones	2
T9	Facilidad de cambio	1	5	Se requiere un costo moderado de mantenimiento	5
T10	Concurrencia	1	0	No hay concurrencia	0
T11	Incluye objetivos especiales de seguridad	1	0	No requiere seguridad	0
T12	Provee acceso directo a	1	2	Existencia de otros módulos con	2

	terceras partes			próxima integración.	
T13	Se requieren facilidades especiales de entrenamiento a los usuarios	1	2	Nueva interpretación de conceptos	2
Total					25

Tabla C.3 Calcular TCF

Factor	Descripción	Peso	Valor	Comentario	Σ ($Peso_i$ * $Valor_i$)
E1	Familiaridad con el modelo de proyecto utilizado	1.5	3	Se tiene cierta familiaridad con el modelo	4.5
E2	Experiencia en la aplicación	0.5	3	Existen cambios en el manejo del lenguaje	1.5
E3	Experiencia en orientación a objetos	1	5	Se domina la programación orientada a objetos	5
E4	Capacidad del analista líder	0.5	4	Tiene experiencias de proyectos anteriores en el tema	2
E5	Motivación	1	5	Altamente motivados	5
E6	Estabilidad de los requerimientos	2	3	Se pueden esperar cambios	6

E7	Personal part-time	-1	2	El trabajo es el mayor tiempo posible	-2
E8	Dificultad del lenguaje de programación	-1	0	Se cuenta con experiencia en el lenguaje	0
Total					24

Tabla C.4 Calcular EF