Universidad de las Ciencias Informáticas

Facultad 9

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

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Alberto Sanchez Maturell

Tutor: Ing. Abel Ernesto Lorente Rodriguez

Ciudad de La Habana, Junio, 2007

Declaración de Autoría.

Declaro que yo, Alberto Sanchez Maturell, soy autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los _____ días del mes de

_____ del año_____.

_____

Firma del autor

Alberto Sanchez Maturell

_____

Firma del tutor

Ing. Abel Ernesto Lorente Rodriguez

Opinión del Tutor.

Título: Análisis de un IDE para múltiples plataformas con tecnologías y herramientas libres para desarrollar software educativo en formato multimedia.

Autor: Alberto Sánchez Maturell

Tutor: Ing. Abel Ernesto Lorente Rodríguez

Luego de realizar previas revisiones del trabajo presentado y teniendo en cuenta las características del tipo de documento reglamentado para este tipo de acto, se puede expresar que el mismo tiene una buena presentación gráfica así como buena estructuración de su contenido, logrando un buen entendimiento de lo expresado en el mismo.

Durante su ejecución su autor cumplió con los objetivos propuestos y se destacó por su alto sentido crítico y responsabilidad. Su independencia, laboriosidad y creatividad también se puso de manifiesto y se creció en el cumplimiento de las tareas planteadas. Es válido señalar que el estudiante paralelamente cumplió exitosamente con la tarea de impartir clases de apoyo como alumno ayudante de ingeniería de software y que además a partir de enero y hasta mayo del presente año lideró al equipo de trabajo demostrando su capacidad de líder.

La selección del método científico, así como su aplicación es muy acertada. La documentación resultante es muy buena en estructura y contenido.

El resultado obtenido puede ser considerado como excelente y de gran utilidad para el desarrollo de una herramienta que permita el desarrollo de multimedia educativa en plataformas libres, además ha demostrado su capacidad y habilidad para resolver problemas propios de la profesión de forma muy adecuada por lo que, junto a las consideraciones expresadas anteriormente, considero que el estudiante está apto para ejercer como Ingeniero en Ciencias Informáticas.

Considero también que los resultados de la investigación deben ser publicados y presentados en eventos científicos.

Teniendo en cuenta lo antes expresado, se propone al Tribunal la calificación de 5 puntos.

_____ _____

Firma del Tutor Fecha

Opinión del Usuario.

El Trabajo de Diploma, titulado “Análisis de un IDE para múltiples plataformas con tecnologías y herramientas libres para desarrollar software educativo en formato multimedia.”, fue realizado en la Universidad de las Ciencias Informáticas (UCI) para la Dirección de Software Educativo de la misma. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface.

□ Totalmente

□ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes:

_____

_____

_____

_____

_____

_____

_____

_____

_____

Y para que así conste, se firma la presente a los ____ días del mes de _____ del año _____

_____	_____
Representante de la entidad	Cargo
_____	_____
Firma	Cuño

Dedicatoria.

Dedicado a mis padres Estrella y Alberto, quienes me infundieron la ética y el rigor que guían mi transitar por la vida. A mi abuela Ana, quien con generosidad dedicó todo su esfuerzo y me impulsó y apoyó a cada momento.

A mis colegas del proyecto, fuente constante de motivación.

Gracias a todos

Alberto

Agradecimientos.

Quisiera agradecer a todas las personas que de una forma u otra han sido mi base de apoyo en el desarrollo de este trabajo, de forma especial:

♦ *A Abel Ernesto Lorente Rodriguez, tutor de este proyecto, por su ayuda y orientación.*

♦ *A mis colegas y equipo de desarrollo Yonnys Pablo Martín, Joan Pablo Jimenez, Roberto Ferrer, Yosber Rodriguez, Anisley de la Caridad, Geyser Zamora, Diana Fernandez por su dedicación en buscar soluciones y alternativas para lograr la idenpendencia tecnológica a la que aspiramos.*

♦ *A Iliana Perez del proyecto SCADA que me ha servido de mucho para orientarme en muchas tareas.*

♦ *A Yailet especialista de software educativo de IP por ayudarme a darle formato a este documento.*

♦ *A mis mejores amigos de la vida, Maikel y Coco por siempre confiar en mí.*

♦ *A mis padres y hermanos por apoyarme a cada instante de mi vida, por el ánimo que me han dado en seguir mi carrera para convertirme en ingeniero.*

De manera muy especial al Comandante en Jefe Fidel Castro Ruz por permitirnos la posibilidad de dar un Clic al Futuro y ser el creador de esta bella obra de la Revolución.

Resumen.

El presente trabajo consiste en el análisis del diseño de un Ambiente de Desarrollo Integrado (IDE) multi-plataforma integrado con herramientas libres, el cual permita desarrollar productos educativos en formato multimedia en la Universidad de las Ciencias Informáticas (UCI) a través de una interfaz amigable para los desarrolladores de software educativos con un alto nivel de fiabilidad.

Esta herramienta propuesta, contribuye al desarrollo eficaz del plan de migración hacia software libre que está implementando la dirección #2 para extender sus productos y mejorar los servicios con fines educativos en formatos multimedia.

El análisis del sistema se basa desde un estudio completo de factibilidad de las herramientas y tecnologías libres que existen con estos fines hasta las características y componentes elementales del sistema incluyendo los procesos correspondientes a la gestión de requerimientos.

Como parte del trabajo se entrevistaron a especialistas de Software Educativos con experiencias y a usuarios avanzados en Software Libre.

Los resultados fundamentales a través de un estudio actualizado de herramientas libres para el desarrollo de multimedia es el análisis del diseño del sistema y la independencia tecnológica ganada con el uso de tecnologías libres.

PALABRAS CLAVES.

IDE, Herramientas Libres, Multimedia, Formatos Libres, Licencias de Software Libre, Alternativas Libres para multimedia, Reproductores Libres, SVG, SWF.

Abstract.

Title: Analysis of an IDE for multiple platforms with technologies and free tools to develop educational software in multimedia format.

The present work consists on the analysis of the design of a Integrated Development Environment (IDE) multi-platform integrated with free tools, which allows developing educational products in format multimedia in the University of the Computer Sciences (UCI) through a friendly interface for the educational software developers with a high level of reliability.

This proposed tool, contributes to the effective development of the migration plan toward free software that is implementing the address #2 to extend their products and to improve the services with educational ends in multimedia formats.

The analysis of the system is based from a complete study of feasibility of the tools and free technologies that exist with these ends until the characteristics and elementary components of the system including the processes corresponding to the administration of requirements.

As part of the work they interviewed to Educational specialists of Software with experiences and advanced users in Free Software.

The fundamental results through an updated study of free tools for the multimedia development are the analysis of the design of the system and the technological independence won with the use of free technologies.

Tabla de Contenido

Índice de Figuras.

Índice de Tablas.

Introducción.

El software evoluciona rápidamente con el tiempo, lo que obliga a ir adquiriendo nuevas versiones si uno no quiere descolgarse de la carrera tecnológica, esto hace que las licencias de los software propietarios sean insostenibles por las políticas constantes de actualizaciones que obligan a los usuarios realizar un gasto importante de dinero para no quedar atrasados rápidamente.

Para organizaciones con un gran número de usuarios, como nuestra universidad el coste llega a ser un factor con un peso muy importante ya que se vende partiendo del número de licencias de uso, y no como una única entidad, y esto no resulta beneficioso para nuestra economía, además el monopolio que existe de tecnologías potentes para la línea de desarrollo de multimedia por ser de empresas norteamericanas no permiten adquirir libremente dichas tecnologías, lo que impide el desarrollo y comercialización de los productos multimedia.

En la UCI el área de producción # 2 está trazando como política un plan de migración a software libre en la producción de software educativo en formatos multimedia que permitirá extender los productos en el mercado con grandes posibilidades de contratos y convenios referentes al desarrollo de los mismos.

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.

Aún es desconocido el camino correcto para llevar la multimedia al terreno de Software Libre y aún más desarrollar los productos con herramientas y tecnologías libres que permitan ofrecer las características necesarias de un ambiente de desarrollo integrado (en adelante, IDE) con interfaz gráfica para usuario final.

Planteamiento del problema de investigación.

La no existencia de un sistema como alternativa libre con las características necesarias para construir productos multimedia que se ejecuten en múltiples plataformas, que además ofrezca una interfaz gráfica amigable y ayude a los usuarios finales a construir cualquier tipo de proyectos educativos en formato multimedia de manera sencilla hace que se retrase gradualmente el plan de migración a Software Libre (en adelante, SWL) que se está implementando en la dirección de producción #2 para el desarrollo de proyectos multimedia.

Para el desarrollo de esa alternativa libre, y evitar el fracaso como lo han tenido otros proyectos con estos fines, es necesario que los desarrolladores cuenten con un análisis y especificación de las funcionalidades a incluir en la interfaz del sistema, y así posibilitar el diseño del IDE e integración de las herramientas que las sustentaría.

El Analista del Sistema se encarga de darle cumplimento a esta situación problemática, el cual "... es responsable del conjunto de requisitos que están modelados en los casos de usos, lo que incluye todos los requisitos funcionales y no funcionales que son casos de uso específico...de delimitar el sistema, encontrando los actores y los casos de uso y asegurando que el modelo de casos de uso es completo y consistente (...). [10]

Para dar solución a la situación problemática existente se tiene que hacer un estudio de las herramientas sobre la que se apoyaria el sistema para ofrecer las salidas en el formato deseado y analizar detalladamente el flujo de trabajo de requerimientos que propone la metodología RUP (Racional Unified Process).

Objeto de estudio.

Análisis de un IDE para múltiples plataformas con tecnologías libres que permita desarrollar software educativo en formato multimedia. Proceso de Captura de Requerimientos del sistema.

Los objetivos de la investigación que se desean alcanzar para darle cumplimiento a este trabajo son:

Objetivo general.

- Definición de un sistema que ofrezca una interfaz comprensiva e integre herramientas libres para el desarrollo de productos educativos en formato multimedia.

Objetivos específicos.

- Definir las tecnologías y herramientas libres que se utilizarán en el desarrollo del IDE.
- Especificar los requerimientos del sistema a desarrollar.
- Diseñar el prototipo de interfaz gráfica del sistema.

Estos objetivos de investigación definen como campo de acción el desarrollo de productos educativos en formato multimedia.

Idea a defender.

El proceso de investigación se basa en el análisis de las herramientas de software libres que ofrezcan algún resultado a la situación problemática que se plantea y éstas puedan integrarse a través de pluging y de esta manera obtener un completo IDE, donde desde el diseño los usuarios finales realicen cualquier producto en formato multimedia, sin la necesidad de preocuparse en aprender a fondo cualquier otra tecnología subyacente.

Si se especifican los requerimientos básicos del sistema detallando cada funcionalidad en forma de casos de uso e implementando su respectivo prototipo de interfaz entonces el IDE dispondría de una especificación de requerimientos que guíe el proceso de desarrollo de software y así la aplicación contará con las características necesarias para ofrecer una solución completa y parte de esa solución sería el resultado en el formato .swf y .svg.

De acuerdo con los objetivos de investigación antes expuestos y la idea a defender que rige la investigación se plantean las siguientes tareas:

Tareas investigativas a desarrollar.

1. Revisar las bibliografías que existen referente a desarrollos de IDE donde ya se haya trabajado en el tema.
2. Realizar un estudio de las posibles herramientas libres para el desarrollo de productos multimedia con tecnologías libres.
3. Analizar el estado del arte del sistema como solución a desarrollar para la producción de software educativo en formato multimedia.
4. Evaluar las alternativas que existen con respecto a formatos multimedia abiertos.
5. Estudiar las Licencias de Software Libre.
6. Definir el entorno del sistema a desarrollar.
7. Modelar la arquitectura del problema de dominio del sistema.
8. Realizar Gestión de requisitos, definidos por requisitos funcionales (RF) y requisitos no funcionales (RNF).
9. Realizar un análisis de las técnicas y herramientas de la ingeniería para la captura de requerimientos.

10. Identificar los casos de uso.
11. Realizar modelo de casos de uso críticos que responden al sistema.
12. Estudiar sobre la herramienta Racional Software Arquitect (RSA).
13. Adquirir y aplicar el lenguaje de modelado estándar 2.0 (UML 2.0).
14. Realizar prototipo de interfaz de usuario.

Para el desarrollo de las tareas científicas se combinan diferentes métodos y técnicas en la búsqueda y procesamiento de la información para que se ejecuten de manera eficaz, además se debe tener en cuenta que la estrategia de investigación es exploratoria.

Los métodos de investigación utilizados son:

Métodos empíricos

- Entrevistas: En el diagnóstico de necesidades, fueron entrevistados a usuarios avanzados de la comunidad de SWL, a profesionales y especialistas en el tema de la producción de multimedia.
- La experimentación: En la elaboración del prototipo no funcional, probando las herramientas para el desarrollo de la interfaz y validando la factibilidad de las que soportan la solución.

Métodos teóricos

- Histórico-lógico: Para conocer, con mayor profundidad, los antecedentes y las tendencias actuales referidas a la propuesta del IDE, sobre todo en lo referente las herramientas; además de conceptos, términos y vocabularios propios del campo.
- Inducción-deducción: Para el estudio de las concepciones y conceptos empleados para el análisis del IDE y sus herramientas; se analizaron otros proyectos similares como F4L, QFlash, permitiendo la extracción de los elementos más importantes por cada uno de ellos.
- Modelación: Para la caracterización de las nuevas funcionalidades que tendrá el IDE cuando al definir los requisitos funcionales.

Resultados esperados.

- Se obtendrá un estudio actualizado de herramientas libres que pueden integrar el IDE para el desarrollo de multimedia. Para llevar a cabo una independencia tecnológica y comercializar nuestros productos.
- Se obtendrá un prototipo no funcional del IDE con todos los componentes y características necesarias para desarrollar aplicaciones multimedia.

El presente documento está estructurado de la siguiente forma:

Estructura del documento

En el **Capítulo 1** se describen los conceptos asociados al software libre y a la ingeniería de software, se ofrece una concepción general de la investigación comparando alternativas de otros sistemas con perspectivas y fines similares.

En el **Capítulo 2** se hace referencia a las tendencias, herramientas y tecnologías existentes en la actualidad que se deben considerar para hacer la selección de aquellas que se van a utilizar en el proyecto.

En el **Capítulo 3** se describe el análisis del sistema, la solución propuesta, la descripción del entorno del sistema. Se modela el problema de dominio, se expone las propiedades y consideraciones del Modelo de Dominio y se describe la Gestión de Requerimientos, la estrategia utilizada, se determinan las funcionalidades del sistema y se describen en detalle.

Capítulo 1

Fundamentación Teórica

Introducción.

La denominada sociedad de la información en la que vivimos requiere cada vez más de mayores cantidades de información soportada en diferentes formatos combinados: texto, imágenes, sonidos y vídeos. Todo ello ha dado lugar a una nueva tecnología, basada en las aplicaciones multimedia: conocerlas y utilizarlas de forma cotidiana es cada vez más necesario, pero la mayoría de las herramientas que ofrecen esta solución son propietarias y es por ello que en la Universidad de las Ciencias Informáticas se está desarrollando una guía o plan de migración en la producción de software educativos en formato multimedia a software libre.

El propósito de este capítulo es sustentar científicamente el trabajo que se presenta y exponer la necesidad de la independencia tecnológica que todos necesitamos.

1.1 Conceptos asociados al software libre.

Para decidir y poder explicar el camino a Software Libre de la multimedia se tienen en cuenta las siguientes definiciones:

1.1.1 Proyecto GNU/Linux.

El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema operativo completo: el sistema GNU. El 27 de septiembre de 1983 se anunció públicamente el proyecto por primera vez en el grupo de noticias net.unix-wizards. Al anuncio original, siguieron otros ensayos escritos por Richard Stallman como el "Manifiesto GNU", que establecieron sus motivaciones para realizar el proyecto GNU, entre las que destaca "retornar al espíritu de cooperación que prevaleció en los tiempos iniciales de la comunidad de usuarios de computadoras".

GNU es un acrónimo recursivo que significa "GNU No es Unix". Stallman sugiere que se pronuncie Ñu (se puede observar que el logo es un ñu) para evitar confusión con

"new" (nuevo). UNIX es un sistema operativo propietario muy popular, porque está basado en una arquitectura que ha demostrado ser técnicamente estable. El sistema GNU fue diseñado para ser totalmente compatible con UNIX. El hecho de ser compatible con la arquitectura de UNIX implica que GNU esté compuesto de pequeñas piezas individuales de software, muchos de los cuales ya estaban disponibles.

Para asegurar que el software GNU permaneciera libre para que todos los usuarios pudieran "ejecutarlo, copiarlo, modificarlo y distribuirlo", el proyecto debía ser liberado bajo una licencia diseñada para garantizar esos derechos al tiempo que evitase restricciones posteriores de los mismos. La idea se conoce en Inglés como copyleft (en clara oposición a copyright, derecho de copia), y está contenida en la Licencia General Pública de GNU (GPL).

Stallman además creó la Free Software Foundation con el objetivo de crear el sistema Unix libre GNU y la potenciación del software libre. El sistema GNU fue diseñado para ser totalmente compatible con UNIX. El hecho de ser compatible con la arquitectura de UNIX implica que GNU esté compuesto de pequeñas piezas individuales de software, muchos de los cuales ya estaban disponibles, como el sistema de edición de textos TeX y el sistema gráfico X Windows, que pudieron ser adaptados y reutilizados; otros en cambio tuvieron que ser reescritos. [1]

La Fundación para el Software Libre (FSF) está dedicada a eliminar las restricciones sobre la copia, redistribución, entendimiento, y modificación de programas de computadoras. Hacemos esto, promocionando el desarrollo y uso del software libre en todas las áreas de la computación, pero muy particularmente, ayudando a desarrollar el sistema operativo GNU. Muchas organizaciones distribuyen cualquier software libre que esté disponible. En cambio, la Fundación para el Software Libre se concentra en desarrollar nuevo software libre, y en hacer de este software un sistema coherente, el cual puede eliminar la necesidad de uso del software privativo o no libre. Además de desarrollar GNU, FSF distribuye copias de software GNU y manuales por un costo de distribución, y acepta donaciones deducibles de impuestos (en los Estados Unidos), para apoyar el desarrollo de software GNU. Muchos de los fondos de la FSF provienen de los servicios de distribución. "Ésta es la razón por la que pedimos que compren CD-ROMs y manuales (pero especialmente CD-ROMs) de la FSF cuando pueda. [2]

1.1.2 Distribuciones GNU/Linux.

Una distribución Linux, o distribución GNU/Linux (abreviada con frecuencia distro) es un conjunto de aplicaciones reunidas que permiten brindar mejoras para instalar fácilmente un sistema Linux (también llamado GNU/Linux). Son 'ramas' de Linux que, en general, se destacan por las herramientas para configuración y sistemas de paquetes de software a instalar.

Existen numerosas distribuciones Linux. Cada una de ellas puede incluir cualquier número de software adicional (libre o no), como algunos que facilitan la instalación del sistema y una enorme variedad de aplicaciones, entre ellos, entornos gráficos, suites ofimáticas, servidores web, servidores de correo, servidores FTP, etcétera.

La base de cada distribución incluye el núcleo Linux, con las bibliotecas y herramientas del proyecto GNU y de muchos otros proyectos/grupos de software. [3]

1.1.3 El software libre.

El software libre es aquel que se encuentra disponible gratuita o comercialmente, se basa en la cooperación y la transparencia y garantiza una serie de libertades a los usuarios. Estas libertades son:

Libertad 0.

Libertad para utilizar el programa para cualquier propósito.

Libertad 1.

Libertad para poder estudiar cómo funciona el programa. Implica acceso al código fuente del mismo.

Libertad 2.

Libertad para redistribuir el programa.

Libertad 3.

Libertad para hacer modificaciones y distribuir las mejoras. Implica también acceso al código fuente del mismo.

En el software libre es posible desarrollar las mejoras o las modificaciones necesarias a las aplicaciones. De este modo, se contribuye a la formación de profesionales en nuevas tecnologías y al desarrollo de la industria nacional.

Por otro lado, todas las mejoras que se realicen no tienen restricciones y se pueden compartir con cualquier otra empresa, institución u organismo que las necesite, al igual

que las propias aplicaciones. En el software propietario, estas mejoras o no se pueden llevar a cabo o quedan en manos de la empresa creadora, que normalmente se reserva los derechos de uso y propiedad intelectual y establece en qué condiciones las comercializará. [4] ***(ver anexo 1).***

1.1.4 Estándares y formatos libres.

El término estándar o Standard, de origen inglés en tecnología: es una especificación que regula la realización de ciertos procesos o la fabricación de componentes para garantizar la interoperabilidad. Son documentos que dan los detalles técnicos y las reglas necesarias para que un producto o tecnología se use correctamente. La normalización o estandarización es la redacción y aprobación de normas. Los estándares de la industria informática pueden aplicarse tanto al hardware como al software.

Se puede entender los estándares como una manera de asegurar que los factores humanos de calidad estarán incorporados en el sistema. Si resumiéramos los beneficios que supone la utilización de estándares podríamos decir que estos favorecen a:

1. Una terminología común. Esto permite que los diseñadores sepan que están discutiendo los mismos conceptos, con lo que se pueden hacer valoraciones comparativas.
2. El mantenimiento y la evolución. Porque todos los programas tienen la misma estructura y el mismo estilo.
3. Una identidad común. Lo que hace que todos los sistemas sean fáciles de reconocer.
4. Reducción en la formación. Los conocimientos son más fáciles de transmitir de un sistema a otro si por ejemplo, las teclas de órdenes están estandarizadas.
5. Salud y seguridad. Si los sistemas han pasado controles de estandarización es difícil que tengan comportamientos inesperados.

Muchos creadores Web sólo prueban sus páginas con Microsoft Internet Explorer, muchas veces por desconocimiento de la existencia y del grado de implantación de otros navegadores o, en ocasiones, simplemente porque no consideran la compatibilidad como un tema importante.

Internet ha creado sus propios mecanismos para evitar este tipo de situaciones. En 1994 se creó el World Wide Consortium (W3), que agrupa a los principales fabricantes de software de Internet, con la misión principal de definir y promover la creación de estándares.

Los formatos libres garantizan interoperabilidad independientemente de la aplicación que utilicen, ya que permiten a cualquier programador desarrollar software que trabaje con estos formatos. Pondremos un ejemplo; si utilizamos Microsoft PowerPoint para una presentación, sólo tienen garantizado el acceso con todas las particularidades del documento los usuarios de este programa.

La restricción en el acceso de la información a un determinado navegador o formato representa una discriminación contra los usuarios de los otros navegadores o aplicaciones y es un hecho que afecta y preocupa especialmente al mundo del software libre. Ya se han aprobado algunos decretos que establecen el cumplimiento de los estándares Web. [5]

1.1.5 Open source.

Una licencia open-source es una licencia que se usa para programas de computadoras, con copyright, que siguen los principios del movimiento Open Source.

Más formalmente, una licencia es considerada Open Source cuando ha sido aprobada por la Open Source Initiative (OSI), donde el criterio lo da la definición de Open Source. El software de dominio público (esto significa sin licencia), cumple todos estos criterios siempre y cuando todo el código fuente esté disponible, y esté reconocido por la OSI y se le permita usar la marca de la misma.

Definición de Open Source.

La Open Source Initiative utiliza la Definición de Open Source para determinar si una licencia de software de computadora puede o no considerarse software abierto.

Bajo la Definición Open Source, las licencias deben cumplir diez condiciones para ser consideradas licencias de software abierto:

- Libre redistribución: el software debe poder ser regalado o vendido libremente.
- Código Fuente: el código fuente debe estar incluido u obtenerse libremente.
- Trabajos derivados: la redistribución de modificaciones debe estar permitida.
- Integridad del código fuente del autor: las licencias pueden requerir que las modificaciones sean redistribuidas solo como parches.

- Sin discriminación de personas o grupos: nadie puede dejarse fuera.
- Sin discriminación de áreas de iniciativa: los usuarios comerciales no pueden ser excluidos.
- Distribución de la licencia: deben aplicarse los mismos derechos a todo el que reciba el programa.
- La licencia no debe ser específica de un producto: el programa no puede licenciarse solo como parte de una distribución mayor.
- La licencia no debe restringir otro software: la licencia no puede obligar a que algún otro software que sea distribuido con el software abierto deba también ser de código abierto.
- La licencia debe ser tecnológicamente neutral: no debe requerirse la aceptación de la licencia por medio de un acceso por clic de ratón o de otra forma específica del medio de soporte del software. [6]

1.1.6 Licencias de software libre.

Una licencia de software (en inglés software license) es el permiso que se le concede al titular del derecho del autor. Existe la Licencia Pública General de GNU que su objetivo principal es garantizar la libertad de compartir y modificar software libre, para asegurar que el software es libre para todos sus usuarios.

Licencias con copyleft. Son aquellas que ceden los derechos de copia, distribución y modificación del programa bajo las condiciones que definen al software libre pero que además exigen que cualquier versión modificada herede el mismo tipo de obligaciones y derechos que tenía el programa original. Estas licencias a menudo se llaman víricas por el efecto de contagio que tienen sobre trabajos derivados. El objetivo es garantizar que cualquier usuario conserve en el futuro las libertades originales que definen al software libre, y este futuro incluye los trabajos derivados del software original.

Licencias de códigos abiertos o permisivos. Aquellas que ceden el uso del programa bajo las condiciones que definen el software libre pero no obligan necesariamente a hacer públicas las mejoras que realicemos sobre el código. Con las licencias más permisivas este tipo alguien puede usar nuestro programa informático libre, hacer ampliaciones y crear un producto propietario sin compartir con el resto de la comunidad las mejoras introducidas. Así por ejemplo, Netscape creó su producto comercial propietario Netscape Directory Server basándose en el código fuente del servidor del software libre de la Universidad de Michigan sin necesidad de tener que

publicar las mejoras, con lo cual sólo los clientes de Netscape podían beneficiarse de estas mejoras a pesar que la empresa había usado un programa libre. [7]

1.2 Conceptos de la ingeniería de software.

Para lograr el desarrollo de los requisitos del sistema se analizaron metodologías, flujos de trabajos y se asignaron roles que interactuarían con el sistema a desarrollar.

1.2.1 Análisis de sistema.

Es la etapa donde se deberán utilizar las herramientas para la recolección de datos como Cuestionarios, Entrevistas, Revisión de Documentos, Observación (a sistemas de mayor, menor o igual complejidad), y representar la información recabada en diagramas de procesos como Yourdon, Hipo, Tablas, Árboles, etc. [9]

Un Análisis de Sistema se lleva a cabo teniendo en cuenta los siguientes objetivos:

Identificación de Necesidades.

Es el primer paso del análisis del sistema. En este proceso el Analista se reúne con el cliente y/o usuario (un representante institucional, departamental o cliente particular), e identifican las metas globales, se analizan las perspectivas del cliente, sus necesidades y requerimientos, sobre la planificación temporal y presupuestal, líneas de mercado y otros puntos que puedan ayudar a la identificación y desarrollo del proyecto.

Este paso es llamado también por otros autores como "Análisis de Requisitos" y lo dividen en cinco partes:

1. Reconocimiento del problema.
2. Evaluación y Síntesis.
3. Modelado.
4. Especificación.
5. Revisión.

Antes de su reunión con el analista, el cliente prepara un documento conceptual del proyecto, aunque es recomendable que este se elabore durante la comunicación Cliente –Analista, ya que de hacerlo el cliente sólo de todas maneras tendría que ser modificado, durante la identificación de las necesidades.

Estudio de Viabilidad.

La viabilidad y el análisis de riesgos están relacionados de muchas maneras, si el riesgo del proyecto es alto, la viabilidad de producir software de calidad se reduce, sin embargo se deben tomar en cuenta cuatro áreas principales de interés:

1. **Viabilidad económica.**

Es la evaluación de los costos de desarrollo, comparados con los ingresos netos o beneficios obtenidos del producto o Sistema desarrollado.

2. **Viabilidad Técnica.**

Es el estudio de funciones, rendimiento y restricciones que puedan afectar la realización de un sistema aceptable.

3. **Viabilidad Legal.**

Es determinar cualquier posibilidad de infracción, violación o responsabilidad legal en que se podría incurrir al desarrollar el Sistema.

4. **Alternativas.**

Otros objetivos:

- Identificar las necesidades del cliente.
- Evaluar los conceptos que tiene el cliente del sistema para establecer su viabilidad.
- Realizar un análisis técnico y económico.
- Asignar funciones al hardware, software, personal, base de datos y otros elementos del Sistema.
- Establecer restricciones de presupuestos y planificación temporal.

1.2.2 El analista de sistema.

A raíz del surgimiento y explotación del uso de las nuevas tecnologías informáticas, la necesidad de organizar aplicaciones, modificar procedimientos, crear nuevos métodos para dirigir los negocios y buscar y aplicar metodologías adecuadas para la creación y desarrollo del producto en tiempo de mercado, nace el Analista de Sistemas.

En este epígrafe se harán referencias y citas a conceptos dados por especialistas a cerca del rol "Analista de Sistema".

Jacobson, Booch y Rumbauch, definen al Analista de Sistema como:

> "el responsable del conjunto de requisitos que están modelados en los casos de usos, lo que incluye todos los requisitos funcionales y no funcionales que son casos de uso específicos. El analista de sistemas es responsable de delimitar el sistema, encontrando los actores y los casos de uso y asegurando que el modelo de casos de uso es completo y consistente. Para la consistencia, el analista de sistemas puede utilizar un glosario para conseguir un acuerdo en los términos comunes, nociones, y conceptos durante la captura de los requisitos. " [10]

Aunque el analista de sistemas es responsable del modelo de casos de uso y de los actores que contiene, no es responsable de cada caso de uso en particular. Esto es una responsabilidad aparte, que pertenece al trabajador Especificador de casos de uso. El analista de sistemas es también el que dirige el modelado y coordina la captura de requisitos.

Hay un analista de sistemas para cada sistema. No obstante, en la práctica, este trabajador está respaldado por un equipo (en talleres o eventos similares) que incluye otras personas que también trabajan como analistas.

Otra definición es la realizada por Ernesto Santos. (1980)

> "…el analista de problemas en computación deberá conocer procedimientos para indagar sobre lo existente y para saber proponer un verdadero sistema racionalizado, pero también deberá conocer sobre modernos sistemas de información, base del diseño, sobre todo en computación… Estos últimos factores son los que justifican tal especialidad, porque realmente debieron existir los analistas de sistemas, aunque no hubiera computadores, toda vez que siempre hubo sistemas para organizar, que posiblemente no se difundieron porque no existieron en importancia esos dos factores que hoy prevalecen: el computador y la información." [11]

La definición de Analista de Sistemas de Senn (1992, p. 12), agrega:

> "…Los analistas hacen mucho más que resolver problemas. Con frecuencia se solicita su ayuda para planificar la expansión de la organización…", es decir, el papel de los analistas sobrepasa los límites impuestos por la definición inicial, también cumplen el papel de asesores, ya sea en sistemas manuales o informatizados, o cualquier otro sistema donde la empresa tenga que invertir en información, después de todo esa es la razón de ser del analista." [11]

1.2.3 Metodologías de desarrollo de software.

Las metodologías para el desarrollo de software describen el conjunto de fases, etapas, actividades y tareas asociadas a la producción de software (aplicación) de calidad. La finalidad del uso de metodologías es lograr el desarrollo de un software de calidad.

Rumbaugh dio la siguiente definición:

> "Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convencionales en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo". [10]

Existen diversos modelos para el desarrollo de software donde cada una tiene ventajas y desventajas, y se utilizan en aquellas situaciones donde un modelo resulta más apropiado que otro.

En el contexto de este proyecto se evaluaron los siguientes modelos:

- **Modelo Espiral.**

  Propuesto inicialmente por Boehm, es un modelo de proceso de software evolutivo, proporciona el potencial para el desarrollo rápido de versiones incrementales del software. Durante las primeras iteraciones, la versión incremental podría ser un modelo en papel o en prototipo. Durante las últimas iteraciones, se producen versiones cada vez más completas del diseño diseñado.

  Este modelo utiliza la construcción de prototipos como mecanismo de reducción de riesgos, permite a quien lo desarrolla aplicar el enfoque de construcción de prototipos en cualquier etapa de evolución del producto. Mantiene el enfoque sistémico de los pasos sugeridos por el ciclo de vida clásico, pero lo incorpora al marco de trabajo iterativo que refleja de forma más realista el mundo real. [12]

- **Modelo Incremental.**

  Este modelo aplica secuencias lineales de forma escalonada mientras avanza el tiempo.

  Se centra en la entrega de un producto operacional por cada incremento, por lo que los primeros incrementos son versiones incompletas del producto final; pero le

proporciona al usuario la funcionalidad que precisa y una plataforma para la evaluación.

El desarrollo incremental es particularmente útil cuando la dotación del personal no está disponible para una implementación completa en la fecha límite establecida para el proyecto. [12]

1.3 Concepción general de las herramientas de autor.

1.3.1 Definición de las herramientas de autor.

Las herramientas de autor (también denominados entornos de autor o lenguajes visuales) son aplicaciones informáticas que permiten elaborar sistemas multimedia. Ofrecen un entorno de trabajo que permite una programación basada en iconos, objetos y menús de opciones, los cuales posibilitan al usuario realizar un producto multimedia (como, por ejemplo, un libro electrónico) sin necesidad de escribir una sola línea en un lenguaje de programación. Los íconos u objetos se asocian a las exigencias del creador, de tal modo que existen íconos para reproducir sonidos, mostrar imágenes (gráficos, animaciones, fotografías, vídeos), controlar dispositivos y/o tiempos, activar otros programas, crear botones interactivos, etc.

Por otra parte, los menús asociados a cada objeto permiten la selección de opciones sin necesidad de recurrir a un lenguaje de programación, al menos para las posibilidades que la propia herramienta incluye directamente. Para programar acciones que son imposibles de realizar mediante los menús de cada objeto, la herramienta de autor posee un lenguaje específico que permite superar las limitaciones o restricciones del programa.

Este tipo de software puede ser aplicado al desarrollo de programas educativos ya que permite la creación de aplicaciones en las que, de forma sencilla y rápida se tiene la posibilidad de cambiar el flujo de la información según las necesidades del alumno, relacionar palabras, incluir cuestionarios y marcadores que evalúen los conocimientos alcanzados, activar animaciones y vídeos explicativos, incorporar sonidos y lenguaje hablado, etcétera. [13]

1.4 Análisis comparativo de otras soluciones.

Las alternativas a lo que es Adobe flash en Linux son escasas, muchos proyectos han estado desarrollándose para que los desarrolladores de multimedia tengan una

alternativa libre, pero ninguno ha dado un resultado a la problemática que se presenta en este trabajo.

F4L *(Flash For Linux)* es un programa que se desarrolló como una propuesta para Linux del Flash, pero siempre ha tenido problemas a la hora de crear flash, aunque actualmente se puede hacer animaciones con él tiene grandes problemas de diseño que no permiten avanzar el proyecto y que impiden que sea una alternativa válida [14], debido a esos problemas este proyecto fue abandonado y se creó el QFlash como propuesta para realizar un clon de Adobe Flash para Linux, esto se logró, principalmente, en la parte de la interfaz, pero aún carente de muchas de sus funcionalidades, solo permite crear animaciones básicas añadiendo textos, formas básicas, botones, etc. [15]

Aunque Qflash no se terminó, logró desarrollar una interfaz muy parecida a la del Flash. Su entorno de desarrollo integra un panel de herramientas, línea de tiempo, inspector de componentes y un editor para ActionScript muy sencillo.

Entre los avances de este proyecto se encuentran:

- Generan películas SWF.
- Permite el dibujo.
- Trabaja con capas.
- Salvan en un formato propio el proyecto, para poder editarlo luego.
- Permite ActionsScript muy básico.

Más tarde, Qflash se unió con F4L para crear un programa conjunto: **Uira.** El proyecto Uira combina los recursos y conocimiento de los proyectos F4L y Qflash, ambos eran aplicaciones código abierto que proveían una alternativa para el software propietario Flash. Uira es un acrónimo para Uira Isn't a Recursive Acronym. Actualmente no se sabe si el proyecto va a continuar debido a la ley DADVSI en Francia. [16]

Hasta ahora los desarrolladores de UIRA han trabajado mucho en el diseño de clases y en la organización del proyecto para evitar problemas como en el F4L, Uira pretende ser un completo IDE sin embargo no han llegado a implementar nada todavía, quedándose en la fase de análisis, aunque si elaboraron un prototipo no funcional bastante sencillo y completo.

En el caso de Ktoon, el objetivo no era realizar un clon de Flash, sino un programa que ofreciera la funcionalidad de Flash en el SO Linux. Ktoon es uno de los programas

colombianos, realizado por la empresa Toonka Films, es Software Libre que funciona bajo plataforma Linux, mediante el que se desea que las personas tengan acceso a un programa que les permita crear animaciones en 2D del tipo cartoons y Anime, lo que hace de Ktoon una aplicación gráfica muy profesional. Ktoon no solo sirve para hacer la labor de dibujo, sino también el render del mismo. [17]

Ktoon está enfocada a la industria de la animación profesional, proyecto cubierto por la Licencia GPL usando G++, OpenGL y QT como recursos de programación desde Kdevelop como plataforma de desarrollo. Por ahora, KToon sólo está disponible para sistemas Unix, como se había mencionado anteriormente, pero esperamos portarlo a otros sistemas operativos también. [18]

Uno de los puntos que se destaca con Ktoon es la claridad de los manuales que están en su página, tanto en español como en inglés, los cuales son muy completos e incorporan una pequeña muestra de animación hecha con base a esos manuales, lo que nos evidencia el poder y practicidad del programa, una vez que se sabe manejar.

Ktoon al tener licencia GPL, permite que haya campo de acción no solo para diseñadores, sino también para los programadores que deseen crear mejoras en el programa, de hecho, es posible bajar el código fuente del mismo. Ktoon no implementa un editor de código ActionScript ni tiene forma de añadírselo a la película por lo que se queda como solo un software para el dibujo y la animación.

Estas soluciones analizadas han sido creadas en lenguajes C++, pero han sido alternativas que aún para los desarrolladores de multimedia y usuarios de software libres no han sido válidas, ya que estos proyectos han tenido muchos problemas como los expuestos anteriormente.

Nuestra propuesta como solución para el desarrollo de productos educativos multimedia será realizada en C++ con Qt4.2, su análisis se apoya en las ideas del Uira y el Adobe Flash, la cual sea capaz de integrar herramientas existentes que son necesarias para generar SWF, dando además la posibilidad de abrir un SVG y convertirlo a SWF, y viceversa.

1.5 Conclusiones.

En este capítulo se ha realizado un análisis general de los conceptos aplicables al software libre para sustentar la investigación y también conceptos de la Ingeniería de Software que se llevará a cabo para el desarrollo del IDE.

Además se ha ofrecido una concepción general de las herramientas de autor, y una comparación de las alternativas existentes para posteriormente hacer el análisis de la propuesta del sistema.

Capítulo 2

Tendencias, Herramientas y Tecnologías.

Introducción.

En la actualidad con el desarrollo de las nuevas tecnologías de la información y las comunicaciones se hace posible un intenso intercambio de ideas, experiencias y conocimientos entre millones de seres humanos. La educación de la población mundial es una compleja y costosísima tarea para las instituciones de este sector y más aún para los países subdesarrollados con pocos recursos tecnológicos.

A nivel mundial existe un consumo de aplicaciones propietarias desarrolladas en otros países fundamentalmente Estados Unidos. Por otra parte el desarrollo de productos también se basan en soluciones dadas con tecnologías privativas extranjeras esto ubica el desarrollo de estas aplicaciones, que son las que mejores profesionales requieren, fuera de cualquier fronteras.

Aún así en el mundo de software educativo y libre se producirá un impacto social visible en el desarrollo de diferentes productos educativos por las mejoras que pueden ocasionar el uso de las herramientas y las tecnologías libres, generando nuevas fuentes de empleo y cambiando de esta manera la visión errónea que han creado en los últimos años empresas y monopolios que han permitido que la tecnología sea posible solo para unos pocos y sea causal de desecho de la fuerza laboral humana.

Según el contexto de los diferentes productos educativos que se desarrollan en la dirección de producción #2 permite lograr fundamentar nuestros desarrollos en el uso de estándares, tecnologías y herramientas libres y licencias con fines comunitarios, permitiendo así que el conocimiento sea libre y no sea posible su uso por unas cuantas minorías, teniendo en cuenta que nos encontramos en una sociedad que basa su poder en el conocimiento, finalmente si el conocimiento está disponible para todos será más leal y competitivo a nivel de comercio la lucha entre mercados y será posible alcanzar mejoras en lo referente a la producción.

2.1 Las tecnologías de la información y comunicaciones (TIC).

Las nuevas tecnologías de la Información y Comunicación son aquellas herramientas computacionales e informáticas que procesan, almacenan, sintetizan, recuperan y presentan información representada de la más variada forma. Es un conjunto de herramientas, soportes y canales para el tratamiento y acceso a la información, constituyen nuevos soportes y canales para dar forma, registrar, almacenar y difundir contenidos informacionales. Algunos ejemplos de estas tecnologías son la pizarra digital (ordenador personal + proyector multimedia), la Web entre otros.

Para todo tipo de aplicaciones educativas, las TIC son medios y no fines. Es decir, son herramientas y materiales de construcción que facilitan el aprendizaje, el desarrollo de habilidades y distintas formas de aprender. Una de las áreas que se ha fortalecido de las TIC es el CSCL (*Computer Supported Cooperative Learning*), Aprendizaje Cooperativo Soportado por Computadora, que basado en teorías de la psicología cognitiva ha creado un área de desarrollo de software y de innovación en Pedagogía. La finalidad es que grupos con el interés común de aprender mejoren las experiencias de interacción entre ellos para consolidar el aprendizaje utilizando a las TIC como medio de coordinación.

A través de las TIC se facilita el surgimiento de nuevos roles en docentes y alumnos, con un nuevo entorno de aprendizaje logrando el acceso a vastos recursos de conocimiento, a colaborar con otros compañeros, consultar a expertos, compartir conocimiento y resolver problemas complejos utilizando herramientas cognitivas.

Las tecnologías de la información y la comunicación tienen varios aspectos que deben tomarse en cuenta sobre todo si se está hablando de las TICs enfocada a la pedagogía. Deben utilizarse dentro de la metodología instrumental de un currículo basado por competencias en la que el uso de las TICs se utiliza como una herramienta en el proceso de enseñanza aprendizaje para la conceptualización de los contenidos. También es importante señalar los diferentes tipos de TICs como las plataformas de enseñanza aprendizaje y el software que se utilizan en las aulas inteligentes todo eso con el servicio de la multimedia, nos da como resultado un impresionante cambio en la calidad de la enseñanza y el aprendizaje. [19]

La tecnología de la Información (TI) está cambiando la forma tradicional de hacer las cosas.

2.2 La tecnología multimedia e informática educativa.

Las Tecnologías Multimedia son el producto de una demanda creciente en lo que a incorporación conjunta y compatibilidad de dispositivos y objetos se refiere, al utilizar herramientas de autor se pueden desarrollar multimedia integrando 3 o más de los datos que son posibles de procesar actualmente por computadora: texto y números, gráficas, imágenes fijas, imágenes en movimiento (animaciones) y sonido. De esta manera se pueden apreciar dos aspectos significativos.

- Las multimedia es la combinación e interacción unívoca de objetos, a través del medio informático, de los diferentes sistemas simbólicos por él movilizados.
- Las multimedia deben ofrecer a los sujetos diferentes itinerario de recorrido de la información, de manera que facilite que no sea un mero receptor pasivo de la información, sino más bien un procesador activo.

Las herramientas de autor que existen permiten al "desarrollador de multimedia" generar los prototipos bajo la técnica llamada "fast prototype" (el método más eficiente de generar aplicaciones). Se reconoce que toda herramienta de autor para este fin eficientizan el proceso de producción de multimedia en la etapa de diseño, la segunda de las cuatro etapas que se reconocen para el desarrollo de la misma, porque allí es donde se digitaliza e integra la información, a continuación se pueden apreciar las etapas del desarrollo de una multimedia.

- Trabajo del autor con quien requiere la aplicación para definirla.
- Diseño de la aplicación.
- Digitalizar la información.
- Integrar la información digitalizada.
- Difusión de la aplicación.
- Soporte a los usuarios.

El desarrollo de Multimedia se auxilia con la tecnología hypermedia la cual permite generar áreas, dentro de una pantalla, sensible al mouse, al toque de una tecla. El sistema permite asociar y explorar cualquier tipo de imagen digitalizada dentro de un programa de cómputo, de modo que el usuario navegue o recorra el programa conforme a sus intereses, regrese a la parte original o se adentre en la exploración de otra parte del programa, sin necesidad de recorrerlo todo. Este sistema de recorrido o

de navegación permite al usuario interactuar con los archivos o partes del programa de acuerdo a sus intereses personales.

Tabla 1. Medios y usos de las aplicaciones multimedia.

MULTIMEDIA EN LOS NUEVOS MEDIOS	a)Como medio de aprendizaje	Por interacción, al ritmo personal, simulando situaciones reales. Con juegos que agilizan habilidades
	b)Como medio informativo	Conectado a bibliotecas electrónicas Accesando información, desde casa, por correo electrónico
LOS USOS DE LA MULTIMEDIA.	1) Medio de orientación. Presentaciones multimedia de índices de orientación en bancos y museos, etc. 2) Medio didáctico. Capacitación *(interactividad y simulaciones)*. Dominio teórico previo a práctica. Posibilita conjugar actitudes y creatividad. 3) Libro electrónico. Mediante el CD-ROM se puede tener acceso a libros y bibliotecas.	

Las ventajas fundamentales que poseen son:

1. Presenta las ventajas comunes a todas las tecnologías, permitiendo además una mayor interacción.
2. Ofrece la posibilidad de controlar el flujo de información.
3. Une todas las posibilidades de la Informática y de los Medios Audiovisuales.
4. Un programa multimedia bien diseñado no corre el peligro de obsolencia, puesto que pueden actualizarse con facilidad los contenidos con pequeños cambios en el software.

Las ventajas pedagógicas se pueden concretar en las siguientes:

1) Mejoran el aprendizaje, ya que el alumno avanza según su propio ritmo individual de aprendizaje.

2) Incrementa la retención de la información en el estudiante, gracias a la interacción que el medio permite entre imágenes, gráficos, textos… y simulaciones de la vida real.
3) Aumentan la motivación del estudiante.
4) Reducen el tiempo de aprendizaje debido a que el alumno puede imponer su propio ritmo de adquisición de información, que la información se hace fácilmente comprensible para el usuario, el que la instrucción se convierte personalizada, y que le refuerzo se hace constante y eficaz.
5) Fuerte consistencia pedagógica.
6) Poseer una metodología homogénea.
7) Ofrecer una evaluación de procesos y no de resultados.

Con la utilización de las multimedia en la educación, sobre todo en los casos en los cuales permitimos que los alumnos se conviertan en diseñadores y constructores de programas, estamos propiciando no sólo la utilización, y en consecuencia el aprendizaje de una tecnología usual en un futuro más o menos cercano, sino también el análisis crítico de la información por el usuario, y su adecuación a las potencialidades del software y a las características de los sistemas simbólicos movilizados en el mismo. [20]

2.3 Descripción de las herramientas.

2.3.1 Análisis del lenguaje de codificación para el desarrollo del IDE.

Se define qué lenguaje se va a utilizar para la codificación evaluando el tipo de proyecto, el tiempo que disponemos, el ciclo de vida del proyecto, así se analiza la herramienta que mejor se adapta a nuestra necesidad.

Evaluación de los siguientes lenguajes de programación: C++ y Java.

C++ es uno de los entornos más serios y poderosos para desarrollar aplicaciones potentes. Desde su versión 4.0 comenzó a tomar mayor popularidad.

Características positivas.

Lenguaje C/C++: C/C++ 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, etc.

Entorno poderoso y liviano: El entorno de Visual C++ es uno de los más poderosos que existe, es estable, intuitivo y ágil.

Depurador poderoso: El depurador integrado de C++ y las herramientas que vienen con el paquete son excelentes para depurar programas.

Otras características muy visibles de C++ son abstracción, el soporte para la programación orientada a objetos y el soporte de plantillas o programación genérica. Por todo esto se puede decir que C++ es un lenguaje multi paradigma que abarca tres paradigmas de la programación: la programación estructurada, la programación genérica y la programación orientada a objetos. [21]

C++

Ventajas:

- 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.
- Es una extensión de C. Por eso, muchos programadores encontrarán muy sencilla la transición, ya que podrán seguir haciendo cosas a la antigua usanza.
- Permite un control de la memoria y una capacidad de programación de bajo nivel impensable en Java.
- 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. [22]

Desventajas:

- Para lograr aplicaciones que se ejecuten en varios SO *(Sistemas Operativos)*, se requiere de cierto esfuerzo mayor que en Java, ya que Java es un lenguaje interpretado y C++ es compilado.
- No presenta una arquitectura estándar de desarrollo orientado a Internet. Digamos que Java es algo más que un lenguaje, es toda una plataforma, y

apoyada por muchas empresas, lo que le otorga un grado de calidad del que carece C++.

- 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.
- 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. Aunque hay muchas librerías en la red para C++, no son estándar del lenguaje, y algunas son de pago.
- Es más complicado de aprender que Java. También Java es complicado, y cuando digo complicado me refiero a programar "bien" en Java, pero te obliga mucho más a seguir una metodología. C++, por ser en parte C, es demasiado libre en ocasiones. [22]

C++ por ser un lenguaje compilado es más rápido que Java, esto hace que al elegir un lenguaje, la necesidad de velocidad de nuestra aplicación inclinaría la balanza hacia C++.

Aunque Java tiene un Framework muy bien diseñado, avanzado y potente existen algunos inconvenientes para desistir, el primero es que, aunque es gratuito, el JDK (Java Development Kit) no es software libre y la intención es construir una aplicación completamente con herramientas libres. Por otro lado, Java no está diseñado para construir aplicaciones rápidas, el hecho de correr bajo una máquina virtual, y utilizar en exceso herramientas como polimorfismo dinámico hacen que las aplicaciones Java tiendan a ser lentas, algo que definitivamente no se desea para la aplicación.

Otro aspecto muy bueno a tener en cuenta es la memoria de la maquina ya que C++ trabaja a bajo nivel y por tanto gestiona mucho mejor el tratamiento de gráficos vectoriales para aplicaciones de este tipo. Además las librerías de Java no están tan desarrollados como para realizar aplicaciones con gráficos vectoriales semejantes a la conocida Adobe Flash, pues como sabemos aplicaciones de este tipo son costosas en cuanto a consumo de recursos y podrían ralentizar mucho una aplicación y más si está basada en una maquina virtual que interpreta bytecode, en cuanto a la librería de contenedores y algoritmos, para vectores, colas y demás, no hay color.

2.3.2 Librerías gráficas para la producción de aplicaciones de interfaz gráficas.

Librería QT.

QT fue creada por la Compañía Noruega Trolltech, es una amplia plataforma que incluye clases, librerías y herramientas para la producción de aplicaciones de interfaz gráfica en C++ aunque han aparecido bindings a otros lenguajes como Python o Perl, además está completamente orientado a objetos, lo que facilita el desarrollo de software, ha sido la base para numerosas aplicaciones incluyendo la popular interfaz gráfica para Linux llamada KDE con un notable éxito y rápida expansión. [23]

Se pueden encontrar las siguientes distribuciones de Qt:

- Qt Enterprise Edition y Qt Profesional Edition, disponibles para el desarrollo de software con fines comerciales. Incluye servicio de soporte técnico y están disponibles ampliaciones.
- Qt Free Edition es la versión para Unix/X11 para el desarrollo de software gratuito y de código abierto. Se puede obtener gratis sujeto a los términos de la Q Public License y de GNU General Public License. Para plataformas Windows también está disponible la versión Qt non comercial.
- Qt Educational Edition es una versión de la Qt Profesional Edition con licencia únicamente para fines educacionales.
- Qt/Embedded Free Edition.

Con la biblioteca QT se pueden desarrollar ricas aplicaciones gráficas y complejas, incluye soporte de nuevas tecnologías como:

- Librerías básicas-> Entrada/Salida, programación para redes, XML, OpenGl.
- Interface con bases de datos-> Oracle, MySQL, PostgreSQL, ODBC.
- Plugins, librerías dinámicas *(Imágenes, formatos,...).*

QT se distribuye bajo una licencia libre GPL*(o QPL)* que nos permite incorporarlas en aplicaciones open-source. Se encuentra disponible para una gran número de plataformas: Linux, MacOs X, Solaris, HP-UX, UNIX con X11. Además, existe también una versión para sistemas empotrados. [23]

Es una biblioteca que se basa en los conceptos de widgets *(objetos)*, Señales-Slots y Eventos *(ej: clic del ratón)*, las señales y los slots son el mecanismo para que unos widgets se comuniquen con otros, los widgets pueden contener cualquier número de hijos, el widget "top-level" puede ser cualquiera, sea ventana, botón, etc y algunos atributos como el texto de etiquetas se modifican de modo similar al lenguaje html.

¿Qué es Trolltech?

Trolltech es una compañía internacional de software que tiene como producto insignia a Qt. Fue fundada en 1994 pero los primeros desarrolladores de Qt comenzaron a trabajar en 1992 siguiendo un desarrollo basado en el código abierto. [24]

Tabla 2. Comparación de bibliotecas gráficas.

Comparación de las librerías gráficas QT y GTK	
QT está escrito en C++, es multiplataforma (cross-platform nature)	**GTK** es esencialmente una interfaz para la programación de aplicaciones orientadas al objeto (API). Aunque está completamente escrito en C, está implementado haciendo uso de la idea de clases y de funciones respuesta o de callback (punteros o funciones).
QT usa Qt Designer para crear la interface gráfica y el entorno de desarrollo Kdevelop, juntas componen un IDE real (similar al de Visual Studio).	**GTK** utiliza Glade, el cual permite el desarrollo rápido y fácil de interfaces que serán utilizadas por las herramientas de GTK, este es un constructor equivalente al QtDesigner.
QT corre (con recompilación) en Linux/Unix, MacOS X y Windows (en todas las versiones).	**GTK** corre en Linux. Aunque existen versiones para Windows.
QT basa sus aplicaciones y el escritorio KDE en "conversaciones" y por la vía DCOP.	**GTK** basa sus aplicaciones y en Gnome utilizan la vía de CORBA.
QT tiene una exhaustiva vinculación con Python y otros lenguajes como Perl and Scheme.	**GTK** tiene una exhaustiva vinculación con C++ (GTKmm), Python (PyGTK) y Ruby. Y menos relación con Perl, Scheme, Lua y otros.

QT tiene una buena documentación y muchos ejemplos provenientes de Trolltech y el equipo KDE.	**GTK** tiene documentación y ejemplos provenientes del equipo GTK.
QT es completamente gratuito para aplicaciones de código abierto,	**GTK** es gratuito (pero no está completamente disponible para Windows y MacOS X).

GTK tiene inconvenientes que en comparación con QT la hacen desechable, en primer lugar, GTK+ solo posee funcionalidad para desarrollar interfaces gráficas de usuario y deja un lugar vacío a aspectos como bases de datos, redes, etc. Por otro lado, el API *(Interfaz de Programación de Aplicaciones)* de GTK+ está escrito de forma estructurada en lenguaje C, y la intención es trabajar de forma orientada a objetos como lo hace QT.

Existen otros paquetes de herramientas multiplataforma para desarrollar aplicaciones gráficas de usuario hecha en C++ como FLTK que nos provee modernas funcionalidades de interfaces de usuario pero está diseñada para ser pequeña y modular, suficiente para ser enlazada estáticamente, pero trabaja bien como librería compartida.

Otra librería gráfica a considerar es wxWidgets, al igual que QT posee funcionalidades para muchos aspectos aparte de la construcción de interfaces gráficas pero tiene algunas desventajas: Su diseño orientado a objetos no es el mejor que uno pueda ver. A veces abusa de la utilización de macros para realizar ciertas operaciones (como las tablas de eventos), aunque esto hace que codificar sea más fácil, también complica la labor de depuración, especialmente para los mismos desarrolladores de wxWidgets. Por otro lado, debido a que el framework se comenzó a desarrollar hace bastante tiempo, no cuenta con soporte para algunas características relativamente modernas del lenguaje C++ como por ejemplo el manejo de excepciones, y la STL.

Todos estos criterios sirvieron de base para elegir QT como la librería ideal por ser Qt una amplia gama de herramientas capaz de facilitar la creación de formas, botones y ventanas de diálogo con el uso del ratón y ser la más gustada y más universal para crear interfaces gráficas de usuario, es la herramienta a utilizar en el desarrollo del IDE, además su extensa librería con clases y herramientas están bien documentadas,

son fáciles de usar y tiene gran herencia de programación orientado a objetos la cual hace de la programación de interfaces gráficas una aventura placentera.

QT cuenta con un sistema de doble licencia, GPL para el desarrollo de software de código abierto como es el caso del sistema a desarrollar, aunque para el desarrollo de aplicaciones comerciales es pago pero se encuentra una gran parte de sus librerías bajo licencia para Windows, otro aspecto novedoso es que actualmente se encuentra en desarrollo QSA *(Qt Scripts for Applications)*, que permite introducir y crear scripts en las aplicaciones creadas con Qt.

El uso de esta librería podría, además, 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.

Se utilizará la versión QT4.2 la cual cuenta con las siguientes características mejoradas:

- Integración de Qt Designer con entornos de desarrollo como Visual Studio .NET, KDevelop e IDEs Xcode.
- Interfaces más modernos para su uso en teléfonos móviles y PDAs.
- Arthur, el nuevo motor de representación gráfica se integrará con dibujado por pixel y vectores. Además Quartz y QuickDraw bajo Mac OS X, Xlib en X11, GDI en Windows, así como PostScript, SVG, y OpenGL, son ejemplos de tecnologías que se espera que se soporten en Qt4.
- Habrá un nuevo motor de representación de texto llamado Scribe con soporte para Unicode y procesamiento de texto WYSIWYG.
- El modelo de vista llamado Interview mostrará un nuevo sistema de representación de archivos mediante tablas (para entenderlo mejor ver esta captura además de las ya conocidas vista de lista y vista en árbol. También se soportarán los idiomas que escriben de derecha a izquierda.
- Nuevas plantillas de clase llamadas Tulip que son más ligeras, seguras y fáciles de usar que los contenedores STL.
- Mejoras en la programación de hilos con conexiones signals-slots a través de hilos o compartición de datos entre hilos de forma segura.

- La API de Qt serán más fáciles de usar y aprender y además simplificará la creación de bindings en otros lenguajes de programación.
- Mejoras en el soporte de accesibilidad, sobre todo con el toolkit ATK de accesibilidad de Sun para UNIX, aunque siguen siendo soportados Accessible en Windows y la infraestructura de accesibilidad en Macintosh.
- Se podrá usar toda la potencia de Qt4 a través de sus clases, temporizadores, accesos a redes, XML, acceso a bases de datos para aplicaciones de línea de comandos que no tengan interfaz gráfica.
- El conjunto de todas las optimizaciones a bajo nivel hechas al código implicará una reducción de tamaño de los ejecutables en un 10%, con un tiempo de inicio un 20% inferior y con un consumo de memoria un 15% menor al de las aplicaciones Qt3. [25]

2.3.3 Librerías gráficas para el tratamiento de imágenes. (Arthur).

Arthur es el sistema de pintado que vamos a utilizar para el tratamiento de imágenes identificado como el framework vectorial de QT, es muy superior a Cairo y aunque sea Cairo una tecnología estándar e independiente solo sirve para Gnome. Arthur tampoco se puede usar en Gnome ya que depende de las QT y de cierta manera limita el desarrollo de KDE, pero QT y KDE son quienes más han innovado en el escritorio GNU/Linux, tienen una calidad mucho mayor y en cuanto a tecnología está bastante más adelantados que GTK/Gnome.

El sistema de pintado Arthur. Arquitectura.

Confrontación entre los avances usados por Qt para el sistema de pintado Arthur.

El Sistema de Pintado de QT4 se basa primordialmente en las clases: QPainter, QPaintDevice, y QPaintEngine.

El QPainter es la clase usada para representar, sacar operaciones como: drawLine () y drawRect ().

El **QPaintDevice** representa un dispositivo que puede pintarse al usar un QPainter; Ambos QWidget y QPixmap son QPaintDevices.

El **QPaintEngine** provee la interfaz que el pintor suele dibujar encima de los tipos diferentes de dispositivos.

Qt4 ha introducido la clase resumen QPaintEngine. Las implementaciones de esta clase proveen la funcionalidad concreta necesitada para acercarse a los carácteres de

imprenta específicos del dispositivo. La clase del QPaintEngine está sólo usada internamente por QPainter y QPaintDevice, y está escondida de los programadores aplicativos, a menos que reimplementen su propio dispositivo para sus subclases del QPaintEngine.

Qt actualmente provee motores de pintura para las siguientes plataformas y APIs:

- Un motor basado en píxeles para la plataforma de Ventanas.
- OpenGL en todas las plataformas.
- La posdata en Linux, Unix, y Mac OS X.
- QuickDraw y CoreGraphics en Mac OS X.

Qt4 logra usar el gradiente. En este caso el Gradiente se usa para describir la transición de un color en un punto dado, y para obtener un color diferente en otro punto. Qt 4 soporta gradientes lineales, radiales, y cónicas.

- Las gradientes lineales son especificadas con un objeto del QLinearGradient.
- Las gradientes radiales son especificados usando un centro, un radio, y un punto focal creando un objeto del QRadialGradient.
- Las gradientes Concal son especificadas usando un centro y un ángulo de principio con un objeto del QConicalGradient.

Qt4 soporta delineación mezclada en alfa y el relleno. El canal de alfa de un color es especificado a través de QColor. El dibujo mezclado en alfa es soportado en Windows, Mac OS, y en sistemas X11 que tienen la extensión de la X Render instalada. Se logra con esta versión de QT abrir un QPainter en un QGLWidget como si fuera un QWidget normal, en versiones previas esto no se podía hacer. Un beneficio enorme de esto es que utiliza la actuación alta de OpenGL para la mayoría de operaciones del dibujo.

En el Sistema de pintado de Qt 4 (Arthur) hace más uso de operaciones nativas de gráficos, ganando beneficios en sus operaciones ya que potencialmente pueden ser realizadas en hardware, dar mejoras significativas de velocidad sobre muchas implementaciones del software.

En Qt4, todos los dispositivos son de amortiguadores doble por defecto. El almacenamiento intermedio es manipulado por QWidget internamente. Otro aspecto muy importante de QT + Arthur es que logra especificar cómo debería estar un contorno lleno. Puede ser un color entero o un QBrush, lo cual lo hace posible a especificar la textura y el gradiente, se llena para ambos el texto y los contornos.

El QImage como un PaintDevice es una gran mejora de Qt4 sobre versiones previas, provee un motor basado en píxeles de pintura de trama que da a usuarios permiso de abrir un pintor en un QImage. El motor de pintura del QImage soporta el set lleno de rasgo de QPainter (los caminos, el antisubmuestreo, alphablending, etc.) Y puede ser usado en todas las plataformas, una ventaja de esto es que se logra garantizar la exactitud de pixel de cualquier operación del dibujo en una forma independiente en la plataforma. La pintura en una imagen es tan simple como dibujar en cualquier otro dispositivo de pintura.

La clase del QImage provee un pixmap independiente que permite acceso directo para los datos de pixel, y puede ser utilizado como un dispositivo de pintura. Qt provee dos clases para datos manipuladores de imagen: El QImage y QPixmap.

- El QImage es diseñado y optimizado para acceso /manipulación directo de pixel.
- El QPixmap es diseñado y optimizado para dibujar.
- El QPicture nos provee una escalabilidad y dibujado vectorial.

La clase del QPaintDevice es la clase de objetos que pueden estar pintados. Un dispositivo de pintura es una abstracción de un espacio de dos dimensiones que puede trazarse usando un QPainter. Las capacidades del dibujo son implementadas por la subclase QWidget, QPixmap, QPicture, QImage, y QPrinter.

El sistema de coordenadas predeterminado de un dispositivo de pintura tiene su origen localizado en la posición dejada en parte superior. La X aumenta para los incrementos de la derecha y hacia abajo. La unidad es un pixel. Hay varias formas para establecer un sistema de coordenadas creado por el usuario usando al pintor, por ejemplo, usando QPainter::SetMatrix ().

Advertencia: Qt requiere que un objeto de QApplicación existe antes de que cualquier dispositivo de pintura pueda ser creado. Los dispositivos de pintura ganan acceso a los recursos de sistema de la ventana, y estos recursos no son inicializados antes de que un objeto aplicativo sea creado.

La clase del QPaintEngine provee una definición abstracta de cómo el QPainter se acerca a un dispositivo dado en una plataforma dada. Qt 4.0 provee varias implementaciones pre-hechas de QPaintEngine, posee además un motor de pintura para cada armazón de la ventana de sistema. [26]

2.3.4 Entorno de desarrollo KDevelop.

KDevelop es un entorno integrado de desarrollo 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 texto Kate. Las características que se mencionan a continuación son específicas del entorno de desarrollo:

- Editor de código fuente con destacado de sintaxis e indentado automático *(Kate)*.
- Gestión de diferentes tipos de proyectos, como Automake, qmake *(para proyectos basados en la biblioteca Qt)* y Ant *(para proyectos basados en Java)*.
- Navegador entre clases de la aplicación.
- Front-end para el conjunto de compiladores de GNU.
- Front-end para el depurador de GNU.
- Asistentes para generar y actualizar las definiciones de las clases y el framework de la aplicación.
- Completado automático del código en C y C++.
- Compatibilidad nativa con Doxygen *(herramienta libre para generar documentación a partir de código fuente).*
- 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 Qt4.2.

- KDevelop puede especificar la dir de instalación de QT en la creación de un proyecto.
- Incluye estilos de uso de QT4.2.
- Presenta nuevas opciones para QT4 en la configuración de proyectos.
- Integración con Qt Designer.
- Mejorado de QMake Manager, entre otras mejoras.

El KDevelop es un IDE para la programación en el entorno KDE, es totalmente gráfico y combinado con el QtDesigner permite realizar aplicaciones con una gran apariencia en poco tiempo. [27]

2.3.5 Compilador ActionScript MTASC.

MTASC es un compilador open source para ActionScript 2.0 capaz de compilar 100 clases en menos de 5 segundos y es más estricto en muchos aspectos en los que el compilador de Flash, se ha creado el compilador teniendo en cuenta el Standard ECMA.

MTASC es independiente de plataforma, se puede compilar para la plataforma que se desee, lo único que se necesita es poder compilar el lenguaje y el compilador de Ocalm.

El compilador funciona desde la línea de comandos y por lo tanto obliga a trabajar con OOP o sea los proyectos tienen que ser un conjunto de clases. Existen dos variantes, se puede escoger trabajar con POO totalmente y exclusivamente con MTASC y la otra es parcialmente con el IDE de Flash y compilar las clases con MTASC.

Si se escoge trabajar totalmente con POO, desarrollar una aplicación desde Linux no se logra ver cómo hacerlo sin depender del IDE de Flash, incluso aunque solo fuese para instanciar la primera clase, sin embargo se puede trabajar con un sistema de clases en la que una de ellas se auto ejecute al empezar la aplicación. [28]

2.3.6 SWFMILL.

Es un XML2SWF2XML con importante funcionalidades, explicándolo de otra forma es un traductor de XML a un swf y los swf en XML, la sintaxis es fácil, sirve para agregar objetos externos al swf que posteriormente usaremos.

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 o haXe, además swfmill puede ser usado para producir estructuras swf tanto complejas como simples. [29]

- Construido basado en los procesadores XSTL (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 las construcción de simples SWFs. [30]

Swfmill es Software Libre, bajo los términos de GNU General Public License (GPL).

2.3.6.1 XML.

XML *(eXtensible Markup Language)*, es un metalenguaje extensible de etiquetas desarrollado por el W3C (). Es una simplificación y adaptación del SGML *(Lenguaje de Marcación Generalizado)* y permite definir la gramática de lenguajes específicos. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML. [8]

Es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

XML permite el almacenamiento de información estructurada en base a un árbol jerárquico de categorías previamente definidas por el usuario.

Ventajas del XML

Es extensible, lo que quiere decir que una vez diseñado un lenguaje y puesto en producción, igual es posible extenderlo con la adición de nuevas etiquetas de manera de que los antiguos consumidores de la vieja versión todavía puedan entender el nuevo formato.

El analizador es un componente estándar, no es necesario crear un analizador específico para cada lenguaje. Esto posibilita el empleo de uno de los tantos disponibles. De esta manera se evitan bugs y se acelera el desarrollo de la aplicación.

Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo. Mejora la compatibilidad entre aplicaciones.

Estructura de un documento XML.

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de pedazos de

información. Además utiliza un conjunto de reglas DTD *(Document Type Definition)* para definir un documento.

Parsers XML.

Un parser o procesador de XML es la herramienta principal de cualquier aplicación XML. Mediante este parser no sólo podremos comprobar si nuestros documentos están bien formados o válidos, sino que también podremos incorporarlos a nuestras aplicaciones, de manera que estas puedan manipular y trabajar con documentos XML.

Actualmente hay muchos y para todos los lenguajes y plataformas: Java, C, Phyton, Visual Basic, Perl, Tcl, Delphi, etc.,

En síntesis XML nos sirve para cualquier tipo de transferencia de datos que queramos hacer. Se puede utilizar para:

- Meta contenido: Describir meta-información sobre otros documentos o recursos en línea.
- Descripción de documentos: Personalizar y enriquecer la descripción de documentos.
- Base de datos: Publicar e intercambiar el contenido de bases de datos.
- Mensajería: Formato para la comunicación entre programas de aplicaciones.

Los archivos codificados en XML serán formateados (para mostrarlos) o transformados utilizando el Lenguaje de de Hoja de Estilo Extensible XSL *(eXtensible Stylesheet Language)*.

2.3.7 Haxe.

El lenguaje de programación haXe es un sistema completamente equipado con un compilador que detectará errores tempranamente en la fase de desarrollo.

Lo que haXe puede hacer es:

- Crear archivos **SWF (Flash)** usando las APIs de Flash para Reproductores 6,7, 8 y 9.
- Generar **código JavaScript** usando la API DHTML del Navegador, tal que puedas crear aplicaciones web con AJAX.
- Generar **ByteCode** que puede ser usado del lado Servidor (usando un plugin Apache) o empaquetado en un ejecutable independiente.

Cada una de esas plataformas tiene su propia API, pero comparten el mismo lenguaje de programación y la misma librería estándar, así, si las clases son código puro (sin usar APIs específicas de una plataforma) ellas pueden ser compiladas y usadas dónde sea, dependiendo de las necesidades.

- Además, haXe hace fácil el **interoperar** entre las diferentes plataformas, proveyendo librerías de protocolos comunes. considerar que aún es posible usar haXe sólo sobre una plataforma si se desea por ejemplo mantener el código actual.
- HaXe tiene un Sistema de Tipado Flexible. No es dinámicamente tipado porque no es capaz de escalar fácilmente y permitir al compilador hacer varios chequeos cuando se modifica la fuente. No es estáticamente tipado así que no hay que asignarle el tipo a toda variable que estemos usando. Siempre se puede deshabilitar el chequeo de tipo cuando sea necesario en lugares particulares del programa.
- HaXe está basado en ECMAScript, entonces es muy familiar para programadores en JavaScript/ActionScript. Incluye poderosas características de otros lenguajes tales como Generics *(sin los problemas relacionados con la implementación Java).*
- HaXe está disponible para Reproductores Flash 6-7-8 y muy pronto el 9. con las características más nuevas de Flash o del sistema que se analiza sin tener necesidad de aprender un nuevo lenguaje cada vez.
- HaXe es muy conservador acerca de la API de Flash. Se puede usar todas la API de Flash más gustado a utilizar en ActionScript.
- Portar código de AS2 a haXe es muy fácil, y se debe notar mejoras adicionales de performance cuando lo ejecutes.
- Se puede escribir alguna parte del código en haXe y usarlo desde una aplicación AS2 pasada de moda.
- HaXe tiene creación de Delegante automática. Esto significa que no tienes necesidad de preocuparte acerca del contexto en el cual tus métodos serán ejecutados. [31]

HaXe es Software Libre, bajo los términos de GNU General Public License (GPL), es el sucesor de MTASC.

2.3.8 SWFTools.

SWFTools son un grupo de herramientas de código abierto para crear y manipular ficheros swf. SWFTools 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.

SWFTools incluye algunas herramientas para convertir el contenido de formatos PDF, JPEG, GIF, WAV y AVI en SWF, y otras para extraer la información y el contenido de ficheros SWF.

Incluye todas las siguientes herramientas:

- **PDF2SWF** Convertidor de PDF a SWF. Genera un fotograma por página del documento PDF. Permite tener texto completamente formateado, incluyendo tablas, fórmulas dentro de la película Flash. Está basado en XPDF (un pdf parser de Derek B. Nooonburg).
- **SWFCombine:** Una herramienta para combinar SWFs, insertar uno dentro del otro.
- **SWFStrings** Explora el SWF en busca de texto.
- **SWFDump** Imprime varias informaciones sobre un SWF determinado
- **JPEG2SWF** Toma un conjunto de fotos y genera una presentación de imágenes (SlideShow) en un SWF.
- **PNG2SWF** igual que JPEG2SWF, pero solo para PNGs.
- **GIF2SWF** Convierte GIFs a SWF. Además puede manejar GIFS animados.
- **WAV2SWF** Convierte archivos de audio WAV a SWFs, usando el codificador L.A.M.E. MP3 encoder library.
- **AVI2SWF** Convierte animaciones AVI en SWF. Soporta compresión Flash MX H.263.
- **Font2SWF** Convierte archivos de Fuentes (TTF, Type1) a SWF.
- **SWFBBox** Permite reajustar el área de un SWF.
- **SWFC** Una herramienta para crear SWF a partir de un script simple.

- **SWFExtract** Permite extraer Movieclips, Sounds, Imágenes etc. de un archivo SWF.
- **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**. [32]

Las herramientas a utilizar de este conjunto son la SWFC encargada de crear SWF resolviendo los inconvenientes que presenta el MTASC con la línea de tiempo y los fotogramas y la herramienta SWFExtract ya que se necesitaría extraer de un SWF los recursos que se necesiten, esta posibilidad se basa en el hecho que los diseñadores en muchas ocasiones entregan para el desarrollo de multimedia este tipo de formato.

2.4 Lenguaje específico para multimedia. ActionScript.

ActionScript es un lenguaje de OOP, utilizado en especial por el entorno Adobe Flash. ActionScript es un lenguaje de script, por tanto no requiere la creación de un programa completo para que la aplicación alcance los objetivos. El lenguaje está basado en especificaciones de estándar de industria ECMA-262, un estándar para Javascript, de ahí que ActionScript se parezca tanto a Javascript.

La versión más extendida actualmente es ActionScript 2.0, que incluye clases y es utilizada en la última versión de Adobe Flash y en anteriores versiones de Flex. Recientemente se ha lanzado la beta pública de Flex 2, que incluye el nuevo ActionScript 3, con mejoras en el rendimiento y nuevas inclusiones como el uso de expresiones regulares y nuevas formas de empaquetar las clases. Incluye, además, Flash Player 8.5, que mejora notablemente el rendimiento y disminuye el uso de recursos en las aplicaciones Adobe Flash.

La inminente versión 3.0 de Actionscript promete equipararse en potencia con lenguajes como Java y C#, gracias a un notable salto evolutivo que le permitirá incrementar la performance y facilitar el desarrollo de complejas aplicaciones orientadas a objetos.

En ActionScript 2.0 se puede usar clases y además una biblioteca de elementos (Un Clip de Película) que puede ser asociado a una clase. Las clases son extensiones del lenguaje ActionScript el cual los programadores pueden escribir ellos mismos, a través de las cuales podrán construir clases o semejanza de las clases de un Clip de Película, las cuales pueden ser usadas para dibujar vectores que se mostrarán dinámicamente. Los ficheros de clases pueden ser usados para programar fácilmente

y estos pueden ser usados para la transferencia entre varios proyectos si es necesario.

Características de la implementación en el Action Script.

- Todas las cosas son diseñadas asincrónicamente, el servicio de llamadas es ubicuo**,** pero los eventos de los objetos no existen.
- La implementación de XML estará presente como lo ha estado desde Flash 5. Enviando y recibiendo XML.

Para ActionScript es necesario ofrecer referencias, completamiento de código y sobresaltado de sintaxis, el código fuente se salva en conjunto con el resto de la película en un formato de archivo como lo hace flash con el .fla. Esto es además algo común para el código ActionScript que es importado desde ficheros de textos externos mediante la inclusión de los mismos. En estos casos los ficheros externos pueden ser compilados con el Motion Tween ActionScript2 Compiler (MTASC).

- Los programadores dicen que el compilador de Macromedia ActionScript 2.0 es demasiado lento, llevándole varios minutos para compilar alrededor de 100 clases, sin embargo el uso del compilador de código abierto MTASC puede ser usado para una compilación mucho más rápida (Situación que cambia con el uso de ActionScript 3.0).
- La sintaxis del ActionScript es muy tolerante a veces haciendo preocupar muy a menudo a los programadores, por lo difícil que resulta la lectura del código. Sin embargo los viejos programadores de ActionScript y JavaScript han encontrado en la versión 2.0 un modelo de clases de programación con la introducción de tipos de datos determinantes que alejan la dificultad del trabajo en versiones previas al lenguaje (sujeto a cambios en ActionScript 3.0).
- La máquina virtual de ActionScript tiende a golpear rápidamente respecto a la cantidad de cómputo que ActionScript puede realizar antes de accionar un descanso interno, especialmente para el reproductor de Flash en Mac.
- A muchas personas no le gusta tener que importar determinadas clases en la herramienta de autor Flash antes de ser usadas por ellos, desafortunadamente para ellos ActionScript 3.0 confía plenamente en la importación de clases haciendo prácticamente imposible el scripting externo.

2.5 Integración de las herramientas en el IDE.

El IDE integraría las siguientes herramientas:

SWFMill: Genera SWF con librería partiendo de XML y elementos externos.
MTASC: inyecta el código necesario.
SWFC: Es una herramienta del paquete SWFTools que genera SWF y permite mediante códigos sencillos crear complejas películas con línea de tiempo y Actionscript, utilizando la librería Ming, la cual esta escrita en C y puede ser utilizada por varios lenguajes para generar SWF.
SWFExtract: Importante herramienta que se debe tener en cuanta porque los diseñadores a veces entregan partes del diseño en el formato SWF, con ella se puede extraer los elementos que la conforman.
Desde un principio el IDE se había concebido usando solamente MTASC, y SWFMill, pero existe el siguiente inconveniente, MTASC es un compilador de ActionScript 2.0, y para no depender solamente de esa versión ya que ha salido otra superior (AS3.0) del lenguaje, se estudió otra alternativa, Haxe.
Otro inconveniente es que estas herramientas resuelven el problema pero a código, y dibujar o generar gráficos y formas se hace complicado. Nuestra propuesta es tener un conjunto de clases en C++ con las que trabajaría el IDE para traducir los eventos que el usuario hace desde la interfaz del IDE y luego este código se traduciría en código ActionScript que sería compilado por el MTASC, obteniendo así una película SWF, pero el MTASC no se usa de esa forma.
Muchos programadores realizan el diseño gráfico *(toda la parte del dibujo)* con la herramienta de autor, y luego crean las clases ActionScript que manejarán todos estos elementos, las compilas y se la inyectan a la película SWF a través del MTASC, como ven no usan el MTASC para el diseño, es puramente para el código.

Otra forma en que se utiliza el MTASC es en conjunto con el SWFMILL si no deseamos utilizar para nada la herramienta Flash para compilar el código y crear el SWF. Se crean las imágenes y gráficos en algún programa de edición de gráficos, luego a través del SWFMILL se genera un swf con estos elementos en la biblioteca, y luego mediante MTASC le añadimos elementos de ActionScript.

Otro problema con el MTASc es que solo se puede insertar código en el primer fotograma de la película.

Para resolver este inconveniente de línea de tiempo, aparece SWFTools importante conjunto de herramientas, dentro de la cual está SWFC, encargada de recibir un

archivo de texto como entrada con un código y genera un SWF, permite varios fotogramas, trabaja con línea de tiempo y la creación de animaciones y dibujo de formas a través de scripts relativamente sencillos, además de otras prestaciones, que no son más que las mismas cosas que se hace con el Flash 8.

¿Cuáles herramientas integraría el IDE?

Usar el SWFMILL para el trabajo con la librería de objetos, o sea al importar los recursos multimedia, se colocan en una librería, esa librería no sería más que un XML que el SWFMILL recibiría y generaría entonces un SWF, capaz de permitir insertar el código luego.

Usar el SWFC para toda la parte del dibujado, animación, gradientes de color, línea de tiempo, etc, siendo este una herramienta principal en nuestro IDE, al insertar un fotograma en la línea de tiempo se generaría un código para los fotogramas. ***(Ver anexo 2)***.

De esta forma se iría generando un script que al ser compilado por el SWFC daría un SWF con una línea de tiempo y con todos los elementos definidos en cada uno de estos fotogramas. Incluso código ActionScript.

¿Dónde queda ahora el MTASC?

El MTASC inyectaría el código a un SWF existente. Cuando el usuario comience a programar sus clases en ActionScript para determinado proyecto en nuestro IDE, se usaría el MTASC para la compilación de todo este código y se inyectaría en el SWF que genera el SWFC junto al SWFMILL.

Por último agregar que la librería MING constituye la base del SWFC, por lo que la estaríamos usando indirectamente, se podría analizar como la usa el Qflash para generar SWF.

2.6 Análisis de tendencias y estándares.

La generación de contenidos gráficos vectoriales interactivos de calidad y con un discreto consumo de recursos complementa un estudio de las tendencias y estándares entre el SWF (ShockWave Flash) y el SVG (Scalable Vector Graphics), son los dos estándares mas implantados en el mundo de la multimedia, la relevancia que adquieren estas tecnologías están dirigidas a la creación de estos contenidos.

El formato SWF se ha convertido en un estándar de facto, aunque está bajo el control de un único fabricante. Cientos de miles de desarrolladores de sistemas hipermedias hacen uso de esta tecnología, ampliamente difundida gracia a una herramienta de autor potente creado por una modesta compañía denominada FutureSplash, el formato fue adquirido en 1997 por la compañía Macromedia como un complemento a su aplicación Director, dedicada a la generación de contenidos multimedia.

Por su parte, SVG es una especificación libre para el desarrollo de gráficos vectoriales elaborados por W3C y basados en XML (*Extensible Markup Language*), un lenguaje cuya aparición fue en 1997. XML es un metalenguaje empleado en la definición de lenguajes basados en etiquetas, como HTML, que permite el almacenamiento de información estructurada en base a un árbol jerárquico de categorías previamente definidas por el usuario.

Tabla 3. Características generales de los formatos SWF y SVG.

Concepto	SWF	SVG
Tipo de especificación	Formato propietario	Especificación libre
Tipo de formato	Binario	Texto ASCII
Formatos	FLA, SWF	SVG
Tipo MIME	application/x-shockwave-flash	image/svg+xml
Compresión	Sí (zlib)	Sí (gzip)
Herramienta de autor	Sí (Macromedia Flash)	Varios (no es imprescindible)
Animación	Sí (Línea de tiempo)	Sí (nodos de interpolación)
Interactividad	Sí (basada en eventos y scripting)	Sí (basada en eventos, SMIL y scripting)
DOM	Propio	Especificación W3C
Modo de visualización	Plug-in (Macromedia Flash Player)	Plug-in (varios)
Streaming	Sí	No

La diferencia fundamental entre los dos formatos radica, aparte de en el tipo de licencia, en el procedimiento de creación y edición. Mientras que SWF es generado a través de Macromedia Flash, una potente herramienta de autor, SVG carece de entornos de desarrollo con prestaciones similares, aunque la situación está cambiando en los últimos meses gracias a la aparición de herramientas como Inkscape o Sketsa. Sin embargo, el formato tipo texto de SVG favorece su edición con un sencillo procesador de textos, independizando su creación de sistemas operativos y plataformas, lo que otorga un mayor control sobre su generación.

En ambos casos, la visualización se lleva a cabo a través de un plug-in del navegador, pero mientras Macromedia Flash Player está instalado por defecto en los browsers más difundidos, como Microsoft Internet Explorer, los plug-ins de SVG están parcialmente implementados, no se distribuyen de forma conjunta con el navegador, y su difusión es por tanto mucho menor.

Se destaca como particularidad que el desarrollo con Flash involucra siempre dos tipos de ficheros distintos: el formato nativo generado por la herramienta de autor con extensión .FLA como solución completa, y el formato comprimido .SWF como parte de esa solución.

Es significativo señalar que muchos de los parámetros de SVG se definen mediante expresiones heredadas de la especificación CSS (Cascading Style Sheets), utilizada junto con HTML para la creación de páginas Web con hojas de estilo. Así ocurre con las unidades de trabajo, el peso de la fuente, la codificación de los colores o las propiedades de bordes y rellenos. Esta circunstancia facilita el aprendizaje de esta tecnología al desarrollador, ya familiarizado con una sintaxis similar. De hecho, es posible modificar el aspecto de las entidades definidas en SVG desde una hoja de estilo externa vinculada al archivo, al igual que se hace en HTML, permitiendo así la anhelada distinción entre contenido y presentación.

Una de las grandes ventajas de SVG es la cuidada atención que presta a procedimientos más propios de la manipulación de imágenes raster (bitmat), como las máscaras, los filtros o la transparencia, en lo que puede considerarse como un intento promovido por el consorcio W3C de crear una especificación que reúna lo mejor de ambas tecnologías. En el caso de la transparencia, se observa cómo mientras en Flash la propiedad radica en el propio espacio de color, en SVG se trata de un atributo más, con lo que tanto la flexibilidad como el rango de variación alcanzados se incrementan notablemente.

Tabla 4. Comportamiento dinámico de los formatos SWF y SVG.

Concepto	SWF	SVG
ANIMACION		
Arquitectura	Línea de tiempo	Objetos de animación (SMIL)
Tipos de animación	Fotogramas/ Interpolada	Interpolada
Gestión de la animación	Automática / Controlada por eventos	Automática / controlada por eventos
Keyframes	Sí	Sí (KeyTimes / KeySplines)
Métodos de creación de animaciones	Movimiento y Morphing	Movimiento y Morphing (limitado)
Tipo de interpolación	Lineal / Polinómica	Discreta / Lineal / Polinómica
Trayectoria definida por el usuario	Sí	Sí
Animación con velocidad variable	Sí	Sí
INTERACCION		
Lenguaje de programación	ActionScript 2.0	ECMAScript
Interfase de programación	ActionScript API	DOM nivel 2
Fuentes de Eventos	Foco / botones / ratón / teclado / cajas de texto /clips de película / fotogramas / ventana	Foco / ratón / visualización / estado / animaciones / modificaciones escena (mutaciones)
Hipervínculo Web	Sí	Sí
Hipervínculo a otras entidades	Sí, Mediante ActionScript	Sí, directamente

La arquitectura adoptada para la generación de animaciones difiere bastante. Mientras que Adobe Flash para generar un SWF emplea la metáfora de la línea de tiempo donde suceden eventos secuencialmente, heredada de los programas de postproducción de video, como Director, SVG descarga en los objetos de animación esta responsabilidad, derivando más el diseño hacia qué se anima en vez de cuándo se anima.

Para la obtención de un comportamiento interactivo, ambos formatos recurren a un lenguaje de programación (ActionScript en Flash y ECMAScript en SVG), que también proceden de un núcleo común como es JavaScript. Resulta destacable la flexibilidad que en SVG desencadenan los eventos de tipo mutación, generados cada vez que se añade, elimina o modifica un nodo de la estructura jerárquica de la escena creada. La posibilidad de establecer vínculos directos entre las entidades definidas en SVG, sin necesidad de acudir a la programación, facilita notablemente la tarea a los diseñadores.

Tabla 5. Integración de recursos multimedia de los formatos SWF y SVG.

Concepto	SWF	SVG
SONIDO		
Frecuencias de sampleo	5.5 khz, 11 khz, 22 khz, 44 khz	No soportado
Tipo	mono / estéreo	No soportado
Streaming	Sí	No soportado
Filtros	Implementados	No soportado
VIDEO		
Tipo	Sincronización audio-video	No soportado
Streaming	Sí	No soportado
Filtros	Implementados	No soportado

SVG no contempla la integración de video y audio en la escena como Flash, se piensa que SVG dará soporte a ambos medios en un futuro próximo. Flash, por el contrario, presenta una excelente integración de diversos contenidos multimedia siendo una potente herramienta de autor, además tiene un formato de archivo robusto, por otro lado tiene el market share que el SVG necesita. [45]

SWF es el formato que más se usa para la entrega de medios de comunicación rico por ser parte de la solución de tan buenas herramientas de autor. El SWF no es un formato abierto como SVG pero existen muchas herramientas para generar y analizar los archivos swf, además es extensible porque permite a terceros extender la funcionalidad básica del formato a través de mecanismos de etiquetas del propio formato.

Se considerará para el IDE la posibilidad de abrir un SVG y convertirlo a SWF y viceversa ya que la aparición de alternativas a prestaciones similares desde el ámbito del software libre está empezando a poner en entredicho la supremacía de los formatos cerrados, SVG ofrece propiedades específicas de diseño muy importantes, adecuadas para el trabajo en aplicaciones relacionadas con el trazado eficaz de planos y esquemas, cuenta además con posibilidad de generación de chaflanes y empalmes, por la definición de tipos de línea o de patrones de sombreado hacen una buena muestra de estas capacidades.

2.7 Recursos multimedia a utilizar en el IDE.

Los recursos multimedia son muy importantes en las animaciones, los usuario de herramientas de autor deben para poder realizar una animación manejar los recursos existentes que en combinación resulta una multimedia, las herramientas de autor permiten poder importar recursos multimedia que previamente necesiten para confeccionar el producto en dicho formato. 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 debe conocer las alternativas que tiene a la hora de manejar cualquier recurso multimedia en el ordenador para el desarrollo de la misma, y entre estas posibilidades, es imprescindible distinguir los distintos tipos de archivos que puede utilizar, así como las distintas herramientas, sus ventajas y sus limitaciones puesto que

para SO como Windows existen potentes herramientas propietarias con excelentes programas que manejan estos formatos, no así para los sistemas derivados de Unix.

Recursos Gráficos:

En el mundo del diseño gráfico existen dos grandes tipos de archivos gráficos: mapa de bits y vectoriales. Por otro lado hay varios formatos gráficos que son estándar y reconocidos por los programas más habituales de cualquier entorno (Windows, Mac OS, Unix) como las imágenes JPG, GIF o TIF.

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 menos resolución espacial y mayor o menos profundidad de color, además un archivo bitmap para guardar la información de un círculo debe guardar todos los puntos de la figura.

Los programas de diseño vectorial manejan de forma primaria archivos del mismo tipo (vectorial). En estos archivos se almacena información de primitivas gráficas, así para guardar la información de un circulo por ejemplo se guarda la posición del centro, el radio, el tipo, color, grosor de la circunferencia y el relleno.

Imágenes rasters (mapas de bits).

Los parámetros que definen la estructura de la imagen en un archivo de mapa de bits son la profundidad de color y la resolución especial.

El concepto de **profundidad de color** hace referencia a la cantidad de información que se guarda para cada elemento de imagen (pixel).

El concepto de **resolución espacial** 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).

Otro parámetro de interés en los archivos bitmap es el sistema de **comprensión** utilizado, existen dos métodos para reducir el tamaño de estos archivos (comprensión): 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 re escalados a un tamaño mayor, pierden calidad. [33]

Imágenes vectoriales.

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 necesitamos la herramienta Flash de Adobe, o de algunas otras como Ineskape (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. [33]

La mayoría de los sistemas gráficos sofisticados (software de animación) utilizan gráficos por vector. La mayoría de los dispositivos de salida (impresoras, monitores, demás) están basados en imágenes por puntos, esto significa que todos los gráficos vectoriales deben ser convertidos a un mapa de bits antes de su salida. Imágenes estáticas de mapas de BIT (JPEG, PNG).

- El formato **JPEG** *(Joint Photographic Experts Group)* es uno de los formatos más utilizado, se trata de un formato bitmap con distintos niveles de comprensión con pérdida de calidad, esto significa que al descomprimir la imagen no obtenemos exactamente la misma imagen que teníamos antes de la comprensión. Su principal ventaja es que permite reducir el tamaño de las imágenes. Permite cualquier resolución especial y cualquier profundidad de bits, por lo que es el más apropiado para la fotografía digital, Los algoritmos de comprensión de JPG tienen una gran difusión. JPEG es un algoritmo diseñado para comprimir imágenes con 24 bits de profundidad o en escala de grises. [33]
- 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 comprensión, resolución y colores de JPEG.

  Es un formato gráfico basado en un algoritmo de comprensión sin pérdida para bitmaps no sujeto a patentes. Este formato fue desarrollado en buena parte para solventar las deficiencias del formato GIF y permite almacenar imágenes con una mayor profundidad de color y otros importantes datos. El método de comprensión utilizado por el PNG es conocido como deflación. [33]

Imágenes estáticas vectoriales (SVG, AI).

– 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 y 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. Es un estándar abierto. [34]

Imágenes animadas de mapas de BIT (GIF, MNG).

– El formato **GIF** *(Grafphics Interchange Format)* de Compuserve junto con el anterior 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 comprensión sin pérdida. El formato GIF es el preferible si se desea que una parte de la imagen sea transparente o si queremos mostrar pequeñas animaciones. [35]

– El formato **MNG** *(Multiple image Network Grafphics)* es un formato de fichero libre de derechos, estrechamente vinculado al formato de imagen PNG, es muy reciente y actualmente no se actualiza tanto como el PNG. Sin embargo varios navegadores como Netscape Navigator y Konqueror ya soportan MNG; y los plugins de MNG están disponibles para Mozilla hasta la versión 1.4, se intenta crear una alternativa a Mozilla, llamada MNGzilla, integrando MNG, otros navegadores son Opera e Internet Explorer.

  Se espera que MNG comience a sustituir a largo plazo al GIF para imágenes animadas en la Red, pero es un sistema complejo y no permite el visionado de una sola imagen cosa que si hace GIF. [36]

Animaciones vectoriales (SVG, SWF, Ktoon).

Recursos de sonidos.

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 se consideran en esta parte utilizar formatos propietarios y abiertos, ya que no existe una definición de soporte bien definida en este sentido.

- El formato **MP3** *(MPEG Audio Layer 3)*, MP3 es la abreviatura, es un formato de comprensión de sonido con perdida (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 comprensión) y las diferencias sutiles entre un audio de CD original y su copia codificada MP3 son difícilmente apreciables. [37]
- El formato **WAV** (o WAVE), es un formato de audio digital normalmente sin compresión de datos desarrollado y propiedad de Microsoft y de IBM que se utiliza para almacenar sonidos en el PC, admite archivos mono y estéreo a diversas resoluciones y velocidades de muestreo, su extensión es .wav, su principal ventaja es que se puede convertir a otros formatos. [38]

Existen otros formatos como **OGG Vorbis, WMA,** etc, con características incluso superiores, pero al analizar los reproductores de Flash actuales vemos que no son capaces de reproducir estos formatos.

2.8 Evaluación de la distribución GNU/Linux a utilizar.

Distribución: Kubuntu para el entorno KDE.

La distribución a utilizar será Kubuntu es una derivación oficial del SO Ubuntu y todos sus paquetes comparten los mismos archivos que Ubuntu. La diferencia radica en el entorno de escritorio. Tal como se explica "El proyecto Kubuntu quiere ser para KDE lo que Ubuntu es para GNOME: Un sistema operativo integrado con todas las características de Ubuntu, pero basado en el entorno de escritorio KDE. Las nuevas versiones de Kubuntu salen de manera regular y predecible; se hace una nueva versión cada vez que sale una actualización de KDE". Aunque esto no es totalmente cierto: En realidad, los lanzamientos son realizados a la par que la distribución madre. [39]

Descripción de las características de Ubuntu:

Ubuntu es una distribución de GNU/Linux que ofrece un sistema operativo predominantemente enfocado a ordenadores de escritorio. Basada en Debian GNU/Linux, Ubuntu concentra su objetivo en la facilidad de uso, la libertad en la restricción de uso, los lanzamientos regulares y la facilidad en la instalación.

Características Generales:

- Ubuntu siempre será gratuito, y no habrá un coste adicional para la «edición profesional»; queremos que lo mejor de nuestro trabajo esté libremente disponible para todos.
- Para hacer que Ubuntu pueda ser usado por el mayor número de personas posible, Ubuntu emplea las mejores herramientas de traducción y accesibilidad que la comunidad del Software Libre es capaz de ofrecer.
- Ubuntu se publica se manera regular y predecible, una nueva versión cada seis meses. Puede usar la versión estable o probar y ayudar a mejorar la versión en desarrollo.
- Ubuntu está totalmente comprometido con los principios de desarrollo del software de código abierto, animamos a la gente a utilizarlo, mejorarlo y compartirlo. [40]

Otras características

- Basada en la distribución Debian.
- Disponible en 4 arquitecturas: Intel x86, AMD64, PowerPC, SPARC (sólo en versión de servidor).
- Los desarrolladores de Ubuntu se basan en gran medida en el trabajo de las comunidades de Debian y GNOME.
- Las versiones estables se liberan cada 6 meses y se mantienen actualizadas en materia de seguridad hasta 18 meses después de su lanzamiento.
- La nomenclatura de las versiones no obedece principalmente a un orden de desarrollo, se compone del dígito del año de emisión y del mes en que esto ocurre. La versión 4.10 es de octubre de 2004, la 5.04 es de abril de 2005, la 5.10 de octubre de 2005, la 6.06 es de junio de 2006 y la 6.10 es de octubre de 2006.
- El entorno de escritorio oficial es Gnome y se sincronizan con sus liberaciones.
- Para centrarse en solucionar raudamente los bugs, conflictos de paquetes, etc. se decidió eliminar ciertos paquetes del componente main, ya que no son populares o simplemente se escogieron de forma arbitraria por gusto o sus bases de apoyo al software libre. Por tales motivos inicialmente KDE no se encontraba con más soporte de lo que entregaban los mantenedores de Debian en sus repositorios, razón por la que se sumó la comunidad de KDE distribuyendo una distribución llamada Kubuntu.

- De forma sincronizada a la versión 6.06 de Ubuntu, apareció por primera vez la distribución Xubuntu, basada en el entorno de escritorio XFce.
- El navegador web oficial es Mozilla Firefox.
- El sistema incluye funciones avanzadas de seguridad y entre sus políticas se encuentra el no activar, de forma predeterminada, procesos latentes al momento de instalarse. Por eso mismo, no hay un firewall predeterminado, ya que no existen servicios que puedan atentar a la seguridad del sistema.
- Para labores/tareas administrativas en terminal incluye una herramienta llamada sudo (similar al Mac OS X), con la que se evita el uso del usuario root (administrador).
- Mejorar la accesibilidad y la internacionalización, de modo que el software esté disponible para tanta gente como sea posible. En la versión 5.04, el UTF-8 es la codificación de caracteres en forma predeterminada.
- No sólo tiene como lazo a Debian el uso del mismo formato de paquetes .deb, Ubuntu tiene uniones muy fuertes con esa comunidad, contribuyendo cualquier cambio directamente e inmediatamente, más que anunciándolos. Esto sucede en los tiempos de lanzamiento. Muchos de los desarrolladores de Ubuntu son también responsables de los paquetes importantes dentro de la distribución de Debian.
- Todos los lanzamientos de Ubuntu se proporcionan sin costo alguno. Los CDs de la distribución se envían de forma gratuita a cualquier persona que los solicite (la versión 6.10 ya no se distribuye en cd's, la versión anterior se sigue distribuyendo sin costo alguno) aquí. También es posible descargar las imágenes ISO de los discos por transferencia directa o bajo la tecnología Bittorrent.
- Ubuntu no cobra honorarios por la suscripción de las mejoras de la "Edición Enterprise". [41]

Existen muchas más distribuciones de GNU/Linux pero enfocan su esfuerzo no sólo en el desarrollo del software libre, sino también en la comercialización de diferentes productos y servicios basados en software de código abierto y propietarios, esto hace que tengan de cierta manera un enfoque comercial, aunque Ubuntu utiliza software privativos sin estar patentizadas es una distribución muy estable actualmente.

En el caso de Ubuntu, seleccionamos la versión Ubuntu Feisty Fawn 7.04, debido a que esta versión además de ser bastante estable contiene mejoras en rendimiento, así como los paquetes y librerías para el desarrollo están mucho más actualizados, incluyendo la última versión del compilador libre para C++: g++, además para esta nueva versión el Kdevelop trae soporte para Qt4.2, permitiendo el desarrollo de nuestro IDE con esa herramienta.

2.9 Evaluación de la licencia de software libre del IDE.

Un programa es software libre si los usuarios tienen todas sus libertades. Así pues, se debe tener la libertad de distribuir copias, sea con o sin modificaciones, sea gratis o cobrando una cantidad por la distribución, a cualquiera y a cualquier lugar sin pedir o pagar permisos.

La libertad para usar un programa significa la libertad para cualquier persona u organización de usarlo en cualquier tipo de sistema informático, para cualquier clase de trabajo, y sin tener obligación de comunicárselo al desarrollador o a alguna otra entidad específica.

La libertad de distribuir copias debe incluir tanto las formas binarias o ejecutables del programa como su código fuente, sean versiones modificadas o sin modificar (distribuir programas de modo ejecutable es necesario para que los sistemas operativos libres sean fáciles de instalar). [42]

En nuestro proyecto se utilizará ``*copyleft*'' para proteger de modo legal estas libertades para todos, ya que exigen la distribución de los trabajos derivados bajo la misma licencia, es decir los términos de distribución no permiten a los redistribuidores agregar ninguna restricción adicional cuando estos redistribuyen o modifican el software. Esto significa que cada copia del software, aún si ha sido modificada, debe ser software libre.

En la actualidad, se debate sobre qué licencia proporciona mayor grado de libertad, se consideran cuestiones complejas como la propia definición de libertad y qué libertades son más importantes. Las licencias copyleft tratan de maximizar la libertad de todos aquellos destinatarios potenciales en el futuro *(inmunidad contra la creación de software privativo)*, mientras que las licencias de software libre sin copyleft maximizan la libertad del destinatario inicial *(libertad para crear software privativo).*

Con un enfoque similar, la libertad del destinatario *(que es limitada por el copyleft)* puede distinguirse de la libertad del propio software (la cual es garantizada por el

copyleft), existe en la actualidad el copyleft fuerte y copyleft débil, pero por las condiciones que el análisis del IDE impone basándose en el estudio de las herramientas y tecnologías libres que las integrará, y los tipos de licencias de esas herramientas, se puede considerar que se utilizará licencias de software libre con copyleft débil la cual permitirá que el software se enlace con módulos no libres.

El "copyleft débil" hace referencia a las licencias que no se heredan a todos los trabajos derivados, ya que se dependerá en algunas circunstancias de la manera en que éstos se hayan derivado. [43]

La licencia que se propone para el IDE es la Licencia Pública General Menor *(LGPL)* (*Lesser General Public Licence*), se aplica a algunos paquetes de software diseñados con características especiales, es bastante diferente de la Licencia Pública General ordinaria ya que esta última se ajusta solamente a la combinación de librerías y herramientas que apliquen criterios de libertad, de cierta manera la LGPL hace menos que la GPL en proteger las libertades del usuario y proporciona a los desarrolladores de programas libres menos ventajas sobre los programas no libres competidores, pero esto trae aparejado que el permiso para usar una librería determinada en programas no libres posibilita a un mayor número de gente a usar una gran cantidad de software libre. [44]

2.10 Valoración de las metodologías y herramientas case a utilizar.

2.10.1 Valoración de la metodología a utilizar. Rational Unified Process (RUP).

La metodología a utilizar es el RUP, 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.

Transmisión, El objetivo es llegar a obtener el release 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:

Ingeniería de Negocios: Entendiendo las necesidades del negocio.

Requerimientos: Trasladando las necesidades del negocio a un sistema automatizado.

Análisis y Diseño: Trasladando los requerimientos dentro de la arquitectura de software.

Implementación: Creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.

Pruebas: Asegurándose que el comportamiento requerido es el correcto y que todo los solicitado esta presente.

Disciplina de Soporte:

Configuración y administración del cambio: Guardando todas las versiones del proyecto.

Administrando el proyecto: Administrando horarios y recursos.

Ambiente: Administrando el ambiente de desarrollo.

Distribución: Hacer todo lo necesario para la salida del proyecto.

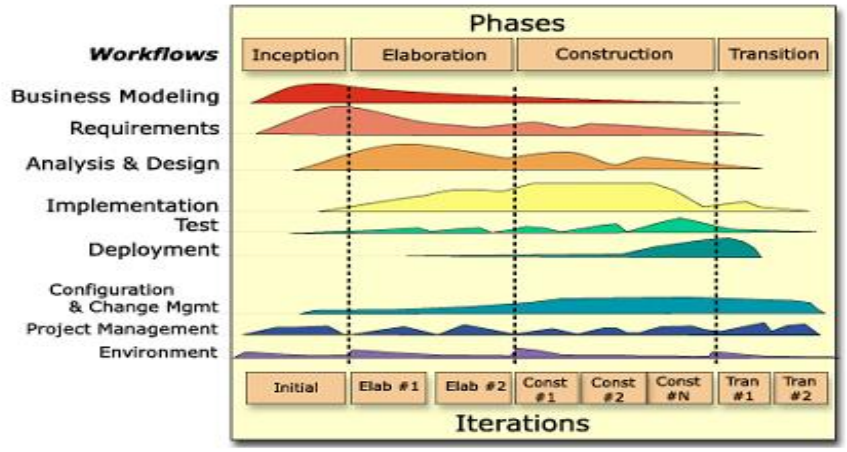

Figura 1. Fases e Iteraciones de la Metodología RUP.

Cada una de estas iteraciones se clasifican y se ordenan según su prioridad, y luego cada una se convierte en un entregable al cliente. Como beneficio se puede apreciar la retroalimentación que se tendría en cada entregable 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. [10]

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.

Otra metodología que se ha analizado es la **MSF** *(Microsoft Solution Framework)* que en consideración con la XP es superior ya que es más flexible con una serie de conceptos, modelos y prácticas de uso, que al igual que el RUP controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos, además se centra en los modelos de proceso, aunque la metodología MSF se compone de varios modelos encargados del desarrollo de un proyecto y se adapta a proyectos de cualquier dimensión, no se puede comparar con las disciplinas que el RUP ofrece.

Como es evidente la metodología del RUP se ajusta a los objetivos de una manera más organizada ya que recoge el aspecto filosófico de la aproximación deseada**,** otro aspecto fundamental es que el RUP se emplea en un entorno amplio de proyectos software donde se analiza la complejidad y el entorno, es aquí una pauta ya que el proyecto software que se desarrolla es de largo plazo y es la metodología más universal.

2.10.2 Análisis del Rational Software Architect (RSA).

La herramienta a utilizar es **RSA** *(Rational Software Architect)* ya que para el desarrollo de Linux esta herramienta ofrece una solución comprensiva que permite

construir, integrar, ampliar, modernizar y desplegar software en Linux. Con la amplia ayuda para todos los roles y actividades en el ciclo de vida del software, la plataforma de desarrollo del software de IBM y los productos de desarrollo Rational apoya el desarrollo para Linux, donde su ayuda se enfoca actualmente en dos áreas:

- Facilitar el desarrollo de aplicaciones de Linux a través de la ayuda comprensiva del IDE que soporta.
- Reducción del coste total de la propiedad para la infraestructura del desarrollo del software proporcionando para Linux la ayuda de la plataforma para la administración de componentes de software.

La familia Rational de productos incluye:

Requisitos y herramientas de análisis. Esta categoría del producto incluye la administración, el modelo de negocio, y las herramientas de modelado de datos.

- IBM RequisitePro Rational: Un producto de gran alcance, fácil de utilizar, e integrado para los requisitos y la gestión de los caso de uso que promueve una comunicación mejor, realza el trabajo en equipo, y reduce riesgo del proyecto.

Herramientas del diseño y construcción. Este grupo de herramientas incluye el desarrollo de soporte integrado estudio premiado de Linux del ambiente del desarrollo de WebSphere (IDE) (ahora nombrado Rational Application Developer). Las herramientas adicionales incluyen modelo de diseño, componente de prueba, análisis de tiempo de ejecución.

- IBM Rational Software Architect: Una herramienta de diseño y de construcción para los arquitectos del software y los principales desarrolladores para la plataforma de Java o en C++ que se apoyan en el desarrollo modelo-conducido con el UML y unifica todos los aspectos de los software aplicados a la arquitectura

El Rational Application Developer proporciona un entorno de desarrollo único y exhaustivo diseñado para cumplir varias necesidades de desarrollo de interfaces Web a aplicaciones del lado del servidor, de desarrollo individual a entornos de equipo avanzados, de desarrollo Java a integración de aplicaciones.

Rational Application Developer es parte de la serie de productos de Rational Software Development Platform. Una plataforma de productos incorporados sobre Eclipse, que es una plataforma de código fuente abierto para crear herramientas de desarrollo de aplicaciones. Cada producto de la familia de escritorio de Rational ofrece el mismo

entorno de desarrollo integrado (IDE.) Las diferencias entre estos productos reflejan qué herramientas de conector están disponibles en cada configuración.

La lista siguiente proporciona una visión general rápida de las características nuevas y de las mejoras de características añadidas:

- Herramientas de portal para desarrollar visualmente aplicaciones de portal.
- Análisis de código automatizado y herramientas de prueba de componentes para mejorar la calidad del código.
- Herramientas de análisis de tiempo de ejecución mejorado para identificar y arreglar problemas de rendimiento al principio del ciclo de desarrollo.
- Herramientas de Crystal Reports incorporadas para construir informes de datos potentes e interactivos.
- Despliegue rápido de WebSphere para acelerar el desarrollo de aplicaciones y simplificar la prueba en WebSphere Application Server.
- Soporte de Eclipse 3.0 para una interfaz de usuario con más interés, más atractiva y personalizable que aumente la productividad del desarrollador.

RSA, a diferencia de otras herramientas MDA ofrece una plataforma abierta para transformaciones sobre Eclipse 3.0, (UML a Java, UML a C++), además una transformación se implementa como un plug-in Eclipse que usa los plug-ins estándar de IBM (UML2, JDT, IBMXTOOLS).

RSA facilita la creación de transformaciones.

- Su motor de transformaciones facilita su programación ofreciendo un modelo orientado al evento (tipo SAX en tratamiento de XML).
- Sus asistentes permiten crear los esqueletos de las transformaciones, etc.
- Proporciona tutoriales (disponible libremente en el repositorio RAS de developerWorks, descargable en RSA mediante su 'RAS browser').

RSA cuenta con los siguientes requerimientos:

- **Procesador** - Mínimo: Pentium™ 3, 800 Mhz; Recomendado: Pentium™ 4, 1.4 GHz.
- **Sistema Operativo:** Windows 2000, XP y Linux.
- **Memoria mínima:** 768 MB; 1 GB de memoria RAM recomendada.

- **Espacio del disco requerido:** 3 GB; 6 GB se requiere al instalar cuando se descarga. [45]

Existen otras herramientas tan buena y completa como RSA, Visual Paradigm que al igual es fácil de usar y de soporte multiplataforma, ambas proporcionan 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 Java y de UML. A pesar de todas estas posibilidades la licencia es muy restringida y Las imágenes y reportes generados, no son de muy buena calidad.

Otra herramienta para implementar es el Umbrello UML Modeller 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, pero Umbrello no es lo suficientemente robusta para un desarrollo tan grande, como es el IDE que se quiere desarrollar para el desarrollo de productos educativos en formato multimedia.

RSA cuenta con licencia de Software Libre de tipo Licencia Publica de IBM, pero es incompatible con la GPL, tiene varios requisitos que no están en la GPL, como por ejemplo: requiere que se den ciertas licencias sobre las patentes.

2.10.3 UML 2 como Lenguaje de Modelado.

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. 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.

Se estableció comparación entre UML 1.5 y UML 2.0; este último ofrece a los proveedores de herramientas CASE *(Computer-Aided Software Engineering)* "la producción automática de programas basado en especificaciones del software".

El UML 1.5 está constituido por 7 diagramas básicos y dos diagramas ***(Ver Figura 2)*** que constituyen variaciones de dos de los anteriores:

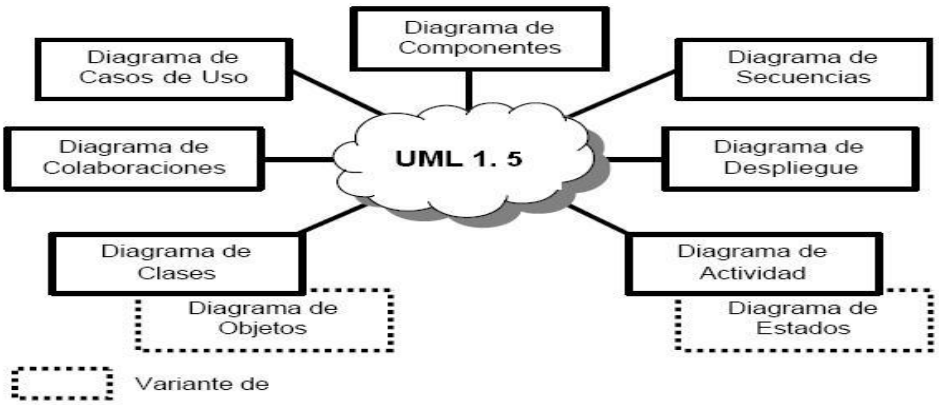

Figura 2. Diagramas de UML 1.5.

En UML 2.0 se definen una serie de diagramas adicionales a los establecidos en UML 1.5. El conjunto de diagramas se encuentra organizado en torno a dos categorías: diagramas estructurales y diagramas dinámicos o de comportamiento. ***(Ver Figura 3)***.

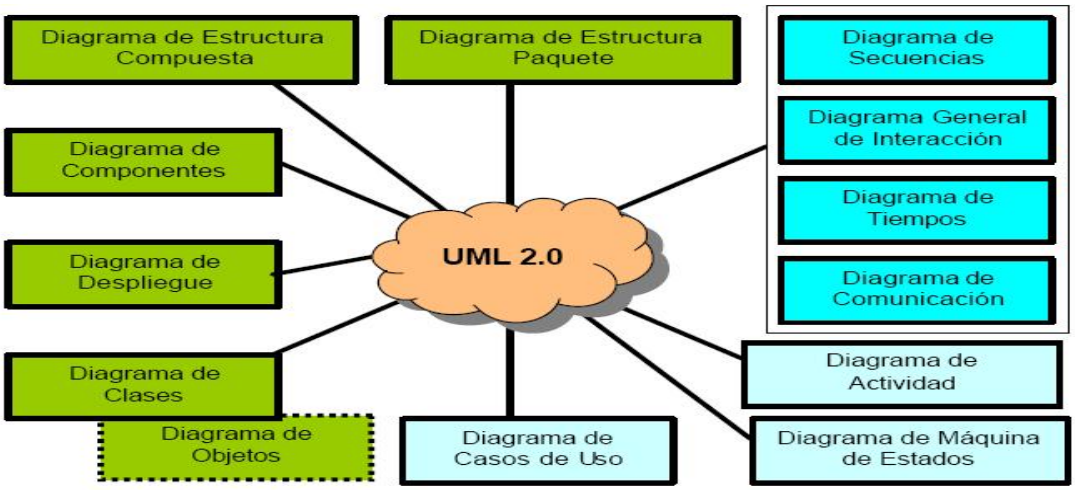

Figura 3. Diagramas de UML 2.0.

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, lazos, entre otros y en el segundo, sobre los diferentes nodos de la infraestructura de red se colocan, a modo de artefactos *(elementos físicos simples)*, los elementos componentes del software. [50]

En UML 2.0 los diagramas aparecen dentro de un marco *(frame)* que posee una etiqueta para indicar el tipo de diagrama, lo que permite al usuario una referencia rápida al diagrama invocado.

Esta versión proporciona a los analistas, arquitectos y desarrolladores; herramientas cada vez más potentes que les permite aprovechar mejor los modelos y como consecuencia generar una mayor cantidad de código reduciendo significativamente el ciclo de desarrollo de sus aplicaciones.

2.11 Eclipse.

Eclipse es un IDE multiplataforma libre para crear aplicaciones clientes de cualquier tipo. La primera y más importante aplicación que ha sido realizada con este entorno es el afamado IDE Java llamado JDT *(Java Development Toolkit)* y el compilador incluido en Eclipse, que se usaron para desarrollar el propio Eclipse.

Eclipse fue creado originalmente por IBM. Ahora lo desarrolla la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

El IDE de Eclipse emplea módulos (en inglés plug-in) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes además de Java. Por ejemplo, existe un módulo para dar soporte a C/C++. Existen módulos para añadir un poco de todo, desde Telnet hasta soporte a bases de datos.

Los componentes gráficos *(widget)* de Eclipse están basados en un juego de herramientas de tercera generación para Java de IBM llamado SWT que mejora los de primera y segunda generación de Sun *(AWT y Swing, respectivamente)*. La interfaz de usuario de Eclipse cuenta con una capa intermedia de interfaz gráfica *(GUI)* llamada JFace, lo que simplifica la creación de aplicaciones basadas en SWT.

Una de sus grandes ventajas es que basa su funcionamiento en plugins con lo que es ampliable para que haga prácticamente cualquier cosa, desde edición de XML a control del Tomcat, pasando por plugins para otros lenguajes como Perl o Shell Script.

En cuanto a las aplicaciones clientes, eclipse provee al programador frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc. [46]

2.12 Conclusiones.

Teniendo en cuenta todas las consideraciones en cuanto a las tendencias, herramientas y tecnologías existentes para el desarrollo de software se decide que el sistema que se propone se desarrolle haciendo uso de las herramientas de software libre, el cual dará la oportunidad de implementar el proceso de migración en la dirección de desarrollo de software educativos con muy bajos costos y con soberanía tecnológica, permitiendo desarrollar e investigar para lograr nuevos avances en esta esfera sin limitaciones.

La herramienta CASE seleccionada para el proceso de modelado acorde al lenguaje UML 2.0, es RSA con un pluging al Eclipse, siguiendo la metodología que propone el RUP. Para el desarrollo de los prototipos de interfaz se utilizó el lenguaje de codificación C++ con la biblioteca gráfica QT 4.2, la cual contiene como framework a Arthur para el sistema de pintado con muy buena arquitectura, El entorno de programación de QT es Kdevelop.

Las herramientas libres que el IDE integrará para obtener el producto requerido, serán el compilador MTASC, al cual se le inyecta el código ActionScript, el SWFmill encargada de generar la librería con elementos externos, HAxe y para el trabajo con la línea de tiempo el SWFC. Todas estas herramientas serán integradas por un sistema de plugin.

Capítulo 3

Descripción de la solución propuesta.

Introducción

En este capítulo se realiza un análisis del sistema a desarrollar, de los componentes y características a utilizar, se descompone el problema de dominio capturando los tipos más importantes de objetos, y se representa a través de un modelo de clases la solución del sistema centrándose en el Proceso Unificado de Desarrollo de Software, haciendo uso de UML *(Unified Model Language).*

Se realiza la gestión de requerimientos a través de diferentes técnicas que existen para la captura de requerimientos, definición de requerimientos y validación de los mismos.

Ha sido de gran utilidad la herramienta Rational Architect Software para el modelado de la aplicación.

Para describir una propuesta de la solución se consideraron primeramente los antecedentes del tema, otros proyectos existentes, de esta manera se han planteado conceptos que aportan mucha claridad en el problema de dominio, así se ha construido el modelo de dominio y se han realizado buenas consideraciones para el entorno del IDE.

3.1 Modelo de dominio.

3.1.1 Descripción del Modelo de Dominio. Glosario de Términos.

La modelación cognitiva que se expone proporciona un nivel de descripción del dominio debido al bajo nivel de estructuración que presenta el negocio que se está estudiando ya que está altamente centrado en herramientas y tecnologías informáticas, se propone un modelo del dominio, permitiendo de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del 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. Este modelo va a contribuir posteriormente a identificar algunas clases que se utilizarán en el sistema.

Se realiza la descripción del modelo de dominio mediante un diagrama de clases UML donde se especifican las principales clases conceptuales que pueden intervenir en nuestro sistema, estos representarán los objetos que existen o eventos que suceden en el entorno en el que trabajará el sistema.

Primeramente se analiza para el dominio el marco de desarrollo de los productos multimedia que se desarrollan en la dirección #2 en la UCI, participando:

- Equipo de diseño instruccional: Definen los contenidos y estructura de información del producto.
- Equipo de diseño visual: Aporta las pautas de interfaz y contenidos visuales tales como realizaciones de diseños, animaciones, ilustraciones.
- Equipo de producción de medias: Produce o gestiona todos los materiales audiovisuales indicados por el equipo de diseño instruccional. Audiciones, sonidos, videos, fotografía.
- Equipo de ingeniería de software: Se encargan de gestionar e integrar todos estos intereses y además implementar en la tecnología seleccionada según los requerimientos funcionales y no funcionales.

Estos equipos aportan al domino una series de elementos y conceptos que se utilizan para la creación de un entorno educativo utilizando elementos abstractos con una

definición explícita. En el dominio intervienen elementos diversos, cada uno de los cuales puede clasificarse en un determinado nivel de abstracción que agrupa diferentes aspectos que intervienen en la definición del entorno del IDE.

Composición: Se denomina composición a la integración de todos los recursos multimedia en el escenario y su resultado será la significación física a un proyecto.

Recursos multimedia: Se entiende por todos los elementos externos o internos para crear una composición.

Elementos internos: Se denomina elemento interno a aquellos objetos gráficos que son creados y manejados por el entorno de trabajo o que lo componen.

Elementos externos: Se denomina elementos externos a aquellos objetos que pueden ser importados por el IDE, llámese imágenes estáticas de mapas de bit, imágenes estáticas vectoriales, imágenes animadas de mapas de bit, animaciones vectoriales, sonidos, videos.

Carácter: Un carácter es un objeto gráfico que posee asociado un estado/instancia con respecto a los fotogramas claves de la línea de tiempo y la escena.

Instancias: Llámese instancia al estado del carácter en el fotograma clave y la escena. Hereda los atributos y eventos de los caracteres para determinar a que fotograma clave u escena está asociado dicho objeto. Las instancias pueden contener código para algún tipo de evento.

Efectos: Se entiende por efectos al comportamiento de los objetos, los efectos modifican dicho comportamiento y se podrán agregar en cualquier tipo de recursos multimedia.

Script: Instrucciones que permiten manejar los recursos multimedia, códigos que son manejados por el controlador de eventos, podrán ser escritos en el lenguaje de ActionScript y estarán orientados direccionalmente.

Controlador Eventos: Clase que se encarga del procedimiento del código que determina qué acciones se ejecutan cuando se produce un evento de algún caracter, como cuando un usuario hace clic en un botón.

Escenas: Componen una composición. Cada escena determina un comportamiento independiente dado por una distribución de capas y caracteres representados desde el punto de vista de la escena.

Biblioteca: Es uno de los componentes de la ventana principal. Se encarga de publicar y organizar los recursos multimedia.

Línea de Tiempo: Se le denomina línea de tiempo al componente del entorno de trabajo ubicado en la parte superior de la pantalla, encima del escenario, compuesto por fotogramas y capas, el cual controla el contenido de las composiciones.

Capas: Se define capa a una película independiente de un único nivel, una capa contiene infinitos fotogramas.

Fotogramas Claves: Se denomina fotogramas claves a los componentes de las capas que controlan y definen el cambio en las propiedades de un objeto de una animación que al incluir un código de ActionScript se puede controlar algunos aspectos del carácter.

Inspector de Propiedades: Componente en forma de ventana. Contiene las paletas de propiedades y eventos aplicables a los recursos multimedia y a otros componentes del entorno de trabajo, dígase escenario, fotogramas.

Editor de Código Script: Se entiende por Editor de código script a la ventana de acciones que edita códigos de ActionScript para los recursos multimedia o un fotograma.

Panel de Herramientas: Componente que contiene herramientas para acelerar operaciones de los elementos internos en el escenario, tales como hacer figuras, además de pintar, seleccionar y modificar ilustraciones, entre otras.

Biblioteca de Componentes Comunes: Componente del entorno de trabajo que contiene un conjunto de caracteres prediseñados/objetos inteligentes.

Paleta de colores: Componente del entorno de trabajo que tiene características para el sistema de pintado de los objetos, fondo de las escenas, insertar código de algún estándar establecido, mezclar los colores.

Ventana Principal: Visualiza todo el entorno de trabajo. Si se cierra, todas los otros componentes (ventanas) también finalizan su servicio. Ella contiene las órdenes relacionadas con el procesamiento de un proyecto concreto o composición.

3.1.2 Diagrama de Clases del Modelo de Dominio.

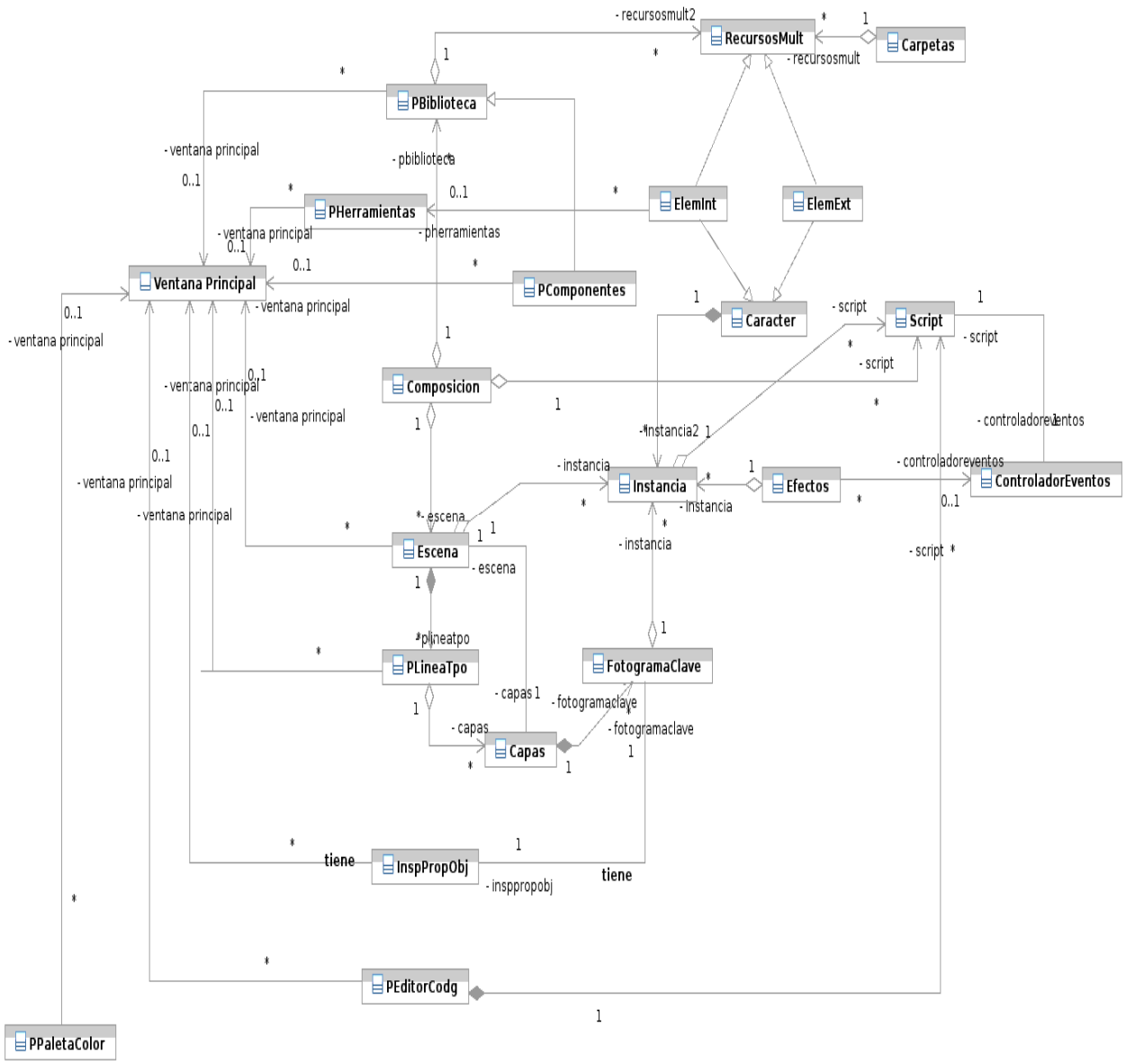

Figura 4. Modelo de Dominio del Sistema.

3.1.3 Propiedades y Consideraciones del Modelo de Dominio.

El IDE trabajará con composiciones, y una composición como se define en el glosario de términos es donde se producirá una animación como resultado, las implementaciones de ellas deben coincidir con las características operaciones del IDE y todos los objetos de una composición se deben poder mover, redimensionar, estar sujetos de efectos, y cada simple pieza debe poder ser exportable a cualquier otro formato.

La clase composición posee la biblioteca donde estarán esos recursos. La biblioteca contiene y puede ser organizada por carpetas ya que los objetos se deben poder obtener desde el esquema de la biblioteca, al importarse un elemento externo o al crear un elemento interno en el escenario, en ambos casos dichos elementos denominados recursos multimedia se agregarán a la biblioteca, y todos ellos serán a la vez una generalización de carácter, provocando que cualquier modificación de un carácter en la biblioteca modificaría todas las ocurrencias que existen.

Los caracteres están asociados a la línea de tiempo. Esta línea de tiempo contiene fotogramas claves y en estos en específicos encontramos a las instancias de los caracteres, las cuales contendrán el código de aquellos caracteres que no sean estáticos. Los caracteres contienen una serie de diferentes atributos que lo diferencian de los otros componentes.

El código script será manejado por el controlador de eventos y los efectos podrán modificar el comportamiento de los objetos y se podrían agregar a cualquier objeto, la clase efecto delega toda su operación a una Instancia de Carácter que posee. El efecto se deriva luego para tener el efecto deseado.

La escena estará organizada en capas. Las capas deberán tener propiedades pero solo contendrán a un carácter, Los caracteres serán como una representación interna que contendrán capas, pero desde el punto de vista de la escena para ella solo será una capa. Eventualmente las capas y caracteres podrán ser mezclados en una sola representación desde el punto de vista de la escena formando un documento.

La escena se compone de Instancias de Caracteres los que contienen el estado en la escena del carácter. Un estado se debe derivar para cada carácter. El carácter posee sus instancias a los Fotogramas clave y puede finalmente estar compuesto de una capa. La escena también posee un Script ejecutable de algunos eventos de la escena.

En cuanto a los caracteres se considera:

- Los caracteres contiene una lista de fotogramas claves, para poderlos asumir básicamente como animación.
- La posibilidad de agregar/quitar un carácter en la animación global, debe ser Mostrar/Ocultar (Un efecto puede hacer eso).
- Cada carácter poseerá un estado, él sería quizás más pequeño que el del carácter instanciado en los fotogramas claves.

- Una composición puede contener la escena múltiple, solamente una escena activa a la vez.

3.1.4 Descripción general del sistema.

GNU/Linux está transformando los entornos corporativos de hoy en día gracias a su consistencia, escalabilidad y bajo coste, Linux es el estándar de facto para muchos aspectos y rápidamente comienza a ser una alternativa a Windows en los PCs de sobremesa. El desarrollo de productos multimedia para Linux con las herramientas tradicionales que existen resulta difícil de usar, ya que no proporcionan la productividad que espera un desarrollador habituado a los rápidos ciclos de desarrollo de las herramientas para Windows.

El IDE a desarrollar debe cumplir con la capacidad del desarrollo rápido de productos multimedia para múltiples plataformas compatible con muchas herramientas de software libre.

La integración de un entorno de desarrollo líder como diseñador visual intuitivo, contará además con un conjunto de clases capaz de analizar y realizar la traducción de las acciones que el usuario realice desde la interfaz, esto se hará a medida que sea necesario sin guardar el resultado de dicha traducción. El IDE estará compuesto por diversas ventanas de herramientas interrelacionadas, tales como explorador de soluciones y el editor de código, de esta manera se podrá manipular aspectos del IDE y responder a sus eventos de manera mas rápida.

No se contará con un reproductor de depuración en sus primeras versiones ya que no contamos con un **Reproductor de SWF libre** que cubra nuestra necesidad porque las alternativas existentes solo permiten reproducir archivos SWF de la versión 7.0 y anteriores, de igual manera no se podrá probar internamente la película debido a que para ello sería necesario pagar una licencia para incluir el Flash Player dentro del IDE, como solución en las opciones del IDE se definirá la ruta al ejecutable del Flash Player 9, compatible para Linux, de esta manera directamente se podrá mandar a reproducir la película con este Reproductor una vez exportada.

El IDE ofrecerá la posibilidad de tener un conjunto de componentes mejorados como pluging para ofrecer el sistema que se necesita en el desarrollo de productos educativos en formato multimedia.

El IDE podrá ser personalizable, ofrecerá distintos métodos para personalizar el espacio de trabajo según sus necesidades. Mediante los paneles y el Inspector de

propiedades se podrá ver, organizar objetos multimedia y modificar sus atributos. Las opciones disponibles en los paneles serán capaces de controlar las características de los símbolos, instancias, colores, tipos, fotogramas y otros elementos.

El IDE permitirá trabajar a través de los paneles con objetos, colores, texto, instancias, fotogramas, escenas y composiciones donde combinando todos estos elementos permitirá al usuario llegar al producto final que desea, existiendo entre ellos una dualidad sincronizada con el código que controla su comportamiento, de tal forma que cualquier modificación en uno de los elementos tendrá su correspondiente e inmediato reflejo en la composición.

El IDE contará con una licencia de software libre compatible con la GPL, se analiza que sea la LGPL o la Guile ya que presenta un copyleft débil por las características que se valoran, la propia FSF *(Free Software Fundation)* recomienda usar la Guile en los mismos casos para los cuales recomienda usar la LGPL.

El IDE debe correr en múltiples plataformas (Linux, Mac, Windows), por lo que la definición de proyecto no puede estar relacionada con alguna característica específica de algún sistema operativo, además debe generar el producto en formato swf el cual será parte de la solución, y también debe ofrecer la posibilidad de abrir un svg y convertirlo en swf y viceversa.

En la actualidad es muy común manejar gran cantidad de información en forma de texto en las multimedia. La utilización y tratamiento de esta información de forma interna en nuestra multimedia trae como consecuencia un sobre gasto de memoria por parte de los reproductores (Ejemplo: FlashPlayer), una solución es utilizar la información de forma externa. El uso de XML es muy práctico para estos casos, por la facilidad de edición de los mismos, así como su fácil elaboración. Nuestro IDE permitirá el trabajo con XML, permitiéndole una mayor facilidad y comodidad al usuario.

El uso de hojas de estilos es una forma muy práctica, cómoda y fácil a la hora de formatear y enriquecer los textos. Su utilización permite ahorrar tiempo de trabajo corrigiendo el formato de los textos.

¿Qué es lo que se quiere que el sistema haga de manera general?

De forma general el usuario debe poder hacer lo siguiente:

- **Importar:** Importar un proyecto, significa abrir una composición desde un fichero, es decir abrir un proyecto, también importar recursos multimedia e importar script.

El usuario debe poder importar un proyecto. Un proyecto es una significación física de una composición. Esto hace que el IDE necesite una definición estable de lo que es un proyecto. La representación de un proyecto debe coincidir con los siguientes requisitos:

- Contener toda la descripción de la composición.
- No ser específico a un sistema operativo.
- Portable (Ej.: Un proyecto creado bajo Linux debe ser transparente bajo Windows).
- Fácil de manejar, es decir las composiciones deben ser fáciles de transportar/enviar por correo.
- El usuario debe poder mover un fichero de una composición a otra.
- Compatibilidad en versiones:
  - Una composición creada bajo la versión 1 debe poder abrirse transparentemente bajo la versión 2.
  - Una composición creada en la versión 2 debe poder abrirse con el mayor esfuerzo bajo la versión 1(con advertencias).

Los recursos multimedia son muy importantes en las composiciones, los usuarios del IDE deberán poder importar cualquier recurso multimedia que previamente tengan y necesiten para su composición. Eventualmente los usuarios deberán poder importar una composición a otra composición considerando esto como un simple objeto de la composición actual.

El IDE ofrecerá características para la edición de script, pero en el mismo sentido se debe dar la posibilidad de que los usuarios editen los script donde ellos deseen, es decir se debe poder ofrecer la posibilidad de importar los script desde otro lugar o desde otra composición (reusabilidad).

- **Autoría:** Hacer lo que se necesita para obtener una composición, crear una composición, manejar los contenidos multimedia e interactivos, controlar el flujo de animación.

El usuario debe poder crear una composición desde cero con el IDE. Diferentes propósitos de proyectos significan diferentes composiciones, la creación de composiciones debe ser completamente transparente.

Los recursos multimedia deben poder ser insertado en la composición cuando el usuario lo desee. Una vez insertado o creado desde la interfaz, los objetos deben poder ser modificables. El contenido multimedia debe obedecer un esquema de la biblioteca, las instancias modifican solamente un subconjunto definido de cada cualidad.

Una animación es una lista de operaciones sobre algún elemento. Las animaciones pueden estar implementadas de manera secuencial y paralela. La primera agrupa operaciones que se ejecutan secuencialmente, es decir, la siguiente operación se ejecuta cuando ha terminado de ejecutar la anterior. Por lo mismo, la duración de esta operación será la suma de las duraciones de las operaciones que agrupa. En cambio, las animaciones agrupadas dentro de un objeto clase paralela se ejecutan todas simultáneamente. Así, la duración será igual a la máxima duración de las operaciones que se agrupan.

Una propiedad importante de las animaciones es que además de ser animaciones, son también operaciones. De esta manera se considera que los flujos de animaciones que se utilizarán en el IDE son: Animación FramebyFrame, es decir cuadro a cuadro, o fotograma a fotograma, esta opción se usará escasamente, también para el desarrollo de una multimedia se utilizará el concepto de Motion Tween, animación de interpolación de movimiento, "rellenar" los fotogramas vacíos de una animación, donde se establece un objeto en el primer frame en una posición y en el último el mismo objeto pero en otra. La interpolación de movimiento calcula donde debería estar el objeto en los frames intermedios.

- **Previsualizar:** Una previsualización del trabajo en progreso pero no dentro del entorno, en esta primera versión se pretende previsualizar el contenido una vez exportado con el flash player 9, ya disponible para Linux.

- **Exportar:** Salvar/Exportar la composición del trabajo en progreso o parte de ella.

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

El IDE debe tener los siguientes componentes:

- Ventana principal

Menú principal (menús comunes a cualquier aplicación).

La barra de programa del margen superior de la pantalla se denominará ventana principal de 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.

El menú principal estará compuesto por los menús de, File, Edit, Insert, Help.

File: 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).

Edit: Es el clásico menú que permitirá Cortar, Copiar, Pegar tanto objetos o dibujos como fotogramas; también permitirá personalizar algunas de las opciones más comunes del programa.

Insert: Te permitirá insertar objetos en la película, así como nuevos fotogramas, capas, acciones, escenas.

Ayuda: Desde aquí se podrá acceder a toda la ayuda, desde el manual existente, hasta el diccionario de Action Script, pasando por tutoriales, lecciones guiadas etc.

- Panel de Herramientas (Acelera las operaciones más comunes del menú principal)

Las herramientas del Panel Herramientas permitirán hacer figuras, además de pintar, seleccionar y modificar ilustraciones. El Panel Herramientas podrá ser personalizable para especificar las herramientas que deben aparecer en el entorno de edición.

- Línea de tiempo (por escena y cada una tendrá varias capas y cada capa una buena cantidad de frame)

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 del documento a través del tiempo en capas y fotograma. Los documentos generados por el IDE dividirán el tiempo en fotogramas. Las capas serán como varias bandas de película

apiladas unas encima de otras, cada una de las cuales contendrá una imagen diferente que aparece en el escenario.

Los componentes principales de la línea de tiempo serán las capas, los fotogramas clave y la cabeza lectora.

Capas: Las capas serán capaces de a organizar las ilustraciones de los documentos. Los objetos de una capa podrán dibujarse y editarse sin que afecten a objetos de otras capas.

Para dibujar, pintar o modificar una capa o una carpeta se deberá seleccionar primero en la línea de tiempo para activarla, sólo puede haber una capa activa en cada momento (aunque se pueda seleccionar más de una capa a la vez).

Inicialmente, un documento contendrá una sola capa. Se podrá agregar más capas para organizar las ilustraciones, la animación y los demás elementos del documento. El número de capas que puedan crearse sólo estará limitado por la memoria del equipo. Las capas no aumentan el tamaño del archivo SWF publicado. El tamaño del archivo sólo se ve afectado por los objetos que se colocan en las capas. También se podrá organizar y administrar capas creando carpetas de capas y colocando las capas en ellas.

Frame: Un frame es un fotograma, una imagen estática, y cuando se pasan muchas rápido dan la sensación de movimiento.

Fotogramas clave: Permitirán producir animaciones sin tener que dibujar cada fotograma por separado, facilitarán la creación de animaciones. Además podrá cambiar rápidamente la longitud de una animación interpolada al arrastrar un fotograma clave en la línea de tiempo, definirá un cambio en las propiedades de un objeto de una animación o incluir un código de ActionScript que controle algunos aspectos del documento.

En la línea de tiempo se trabajará con fotogramas clave colocándolos en el orden en que deben aparecer los objetos de los fotogramas, además permitirá ver los fotogramas interpolados de una animación.

La cabeza lectora: Es lo que determina qué fotograma de nuestra película se está reproduciendo en cada momento.

- Panel Inspector de propiedades de Objetos (contiene las paletas de propiedades y eventos aplicables al objeto seleccionado)

Como se ha mencionado, los objetos vienen definidos por sus propiedades, y los eventos ante los que reaccionan (aparte de los "métodos" de que disponga, que son parecidos a los procedimientos).

El inspector de propiedades de objetos es una ventana desde la cual se podrá ver y modificar la mayoría de las propiedades y eventos de objetos simplificando al máximo la creación de objetos y facilitando el acceso a los atributos más utilizados en la selección actual, ya sea en el escenario o en la línea de tiempo, sin necesidad de acceder a los menús o los paneles que controlan dichos atributos.

Muestra información y la configuración del elemento que está seleccionado, que puede ser un texto, un símbolo, una forma, un mapa de bits, un vídeo, un grupo, un fotograma o una herramienta. Además el inspector de propiedades mostrará el número total de objetos seleccionados.

- 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.

A medida que el usuario realiza su trabajo si es necesario podrá personalizar la vista del escenario la cual dependerá de la resolución del monitor y del tamaño del documento que se desarrollará.

- Panel Biblioteca.

El panel Biblioteca es donde se guardarán y organizarán los símbolos creados en el IDE, además de archivos importados tales como gráficos de imágenes de mapa de bits, archivos de sonido y clips de vídeo. También permitirá organizar los elementos de biblioteca en carpetas.

- Panel de componentes

El panel Componentes permitirá acceder a los Componentes ya construidos y listos para ser usados que proporcionará el IDE. Los componentes son objetos "inteligentes" con propiedades características y muchas utilidades.

- Editor de código script

El panel Editor de código script permitirá crear y editar códigos de ActionScript para un objeto o un fotograma. Se activará este panel al seleccionar un fotograma, botón o instancia de clip de película.

- Panel Mezclador de Colores.

Este panel se utilizará para seleccionar una mezcla de color determinado, se podrá modificar el color de un borde, modificar relleno, también se permitirá introducir código del color según el estándar establecido.

3.2 Gestión de requerimientos. Ingeniería de requerimientos

Cuando se plantea un proyecto hay varias cosas a tener en cuenta antes de decidir la técnica que se va a usar: los requerimientos de sistema, el tiempo de producción, el presupuesto, el mantenimiento y los usuarios a quienes va dirigido, desde el punto de vista de la herramienta/técnica, se desarrollan cada uno de estos aspectos.

La idea es describir la estructura del "sistema" con modelos que describan la visión que tienen los distintos interesados, pero antes es necesario hacer un análisis de los requisitos ya que ellos especifican las características operacionales del software.

Es importante recalcar que estas perspectivas son las que tienen los interesados del sistema, de esta manera, habrán intereses en el **dominio del problema** y su funcionamiento, las **aplicaciones** y sus relaciones, la **información** (los datos) que maneja el área de producción a quien estará dirigido el sistema y la **tecnología** (hardware y software de base) que da sustento informático.

De esta manera se puede considerar que la ingeniería de requisitos del software es un proceso de descubrimiento, refinamiento modelado y especificación.

La definición de las necesidades de un sistema es un proceso complejo, pues en él hay que identificar los requisitos que se deben cumplir para satisfacer las necesidades de los usuarios finales y de los clientes.

El proceso de especificación de requisitos se puede dividir en tres grandes actividades:

1- Captura de requisitos.

2- Definición de requisitos.

3- Validación de requisitos.

A continuación se explicarán brevemente algunas técnicas clásicas para realizar las actividades expuestas anteriormente.

3.2.1 Captura de requisitos.

La captura de requisitos es la actividad mediante la cual el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema.

El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa.

Las técnicas siguientes, de forma clásica han sido utilizadas para esta actividad en el proceso de desarrollo de todo tipo de software.

Entrevistas: resultan una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Existen muchos tipos de entrevistas y son muchos los autores que han trabajado en definir su estructura y dar guías para su correcta realización.

Básicamente, la estructura de la entrevista abarca tres pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados (protocolo de la entrevista).

A pesar de que las entrevistas son esenciales en el proceso de la captura requisitos y con su aplicación el equipo de desarrollo puede obtener una amplia visión del trabajo y las necesidades del usuario, es necesario destacar que no es una técnica sencilla de aplicar. Requiere que el entrevistador sea experimentado y tenga capacidad para elegir bien a los entrevistados y obtener de ellos toda la información posible en un período de tiempo siempre limitado. Aquí desempeña un papel fundamental la preparación de la entrevista.

JAD (Joint Application Development/ Desarrollo conjunto de aplicaciones): esta técnica resulta una alternativa a las entrevistas. Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes. Está basada en cuatro principios

fundamentales: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación "lo que ve es lo que obtiene" *(WYSIWYG, What You See Is What You Get)* es decir, durante la aplicación de la técnica se trabajará sobre lo que se generará. Tras una fase de preparación del JAD al caso concreto, el equipo de trabajo se reúne en varias sesiones. En cada una de ellas se establecen los requisitos de alto nivel a trabajar, el ámbito del problema y la documentación.

Durante la sesión se discute en grupo sobre estos temas, llegándose a una serie de conclusiones que se documentan. En cada sesión se van concretando más las necesidades del sistema.

Esta técnica presenta una serie de ventajas frente a las entrevistas tradicionales, ya que ahorra tiempo al evitar que las opiniones de los clientes se tengan que contrastar por separado, pero requiere un grupo de participantes bien integrados y organizados.

Brainstorming *(Tormenta de ideas)*: es también una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo diez personas), una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador.

Como técnica de captura de requisitos es sencilla de usar y de aplicar, contrariamente al JAD, puesto que no requiere tanto trabajo en grupo como éste. Además suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que suele aplicarse en los primeros encuentros.

Concept Mapping *(Mapas Conceptuales)*: son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar. Son muy usados dentro de la ingeniería de requisitos, pues son fáciles de entender por el usuario, más aún si el equipo de desarrollo hace el esfuerzo de elaborarlo en el lenguaje de éste. Sin embargo, deben ser usados con cautela porque en algunos casos pueden ser muy sugestivos y pueden llegar a ser ambiguos en casos complejos, si no se acompaña de una descripción textual.

Sketches y Storyboards *(Esbozos y Guiones Gráficos)*: Está técnica es frecuentemente usada por los diseñadores gráficos de aplicaciones en el entorno Web. La misma consiste en representar sobre papel en forma muy esquemática las diferentes interfaces al usuario *(esbozos)*. Estos esbozos pueden ser agrupados y unidos por enlaces dando idea de la estructura de navegación *(guiones gráficos).*

Casos de Uso: Aunque inicialmente se desarrollaron como técnica para la definición de requisitos, algunos autores proponen casos de uso como técnica para la captura de requisitos. Los casos de uso permiten mostrar el contorno *(actores)* y el alcance *(requisitos funcionales expresados como casos de uso)* de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Los actores son elementos externos *(personas, otros sistemas.)* que interactúan con el sistema como si de una caja negra se tratase. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores.

La ventaja esencial de los casos de uso es que resultan muy fáciles de entender para el usuario o cliente, sin embargo carecen de la precisión necesaria si no se acompañan con una información textual o detallada con otra técnica como pueden ser los diagramas de actividades.

Cuestionarios y Checklists *(Listas de Chequeo)*: Esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario *(listas de chequeo)*. Este cuestionario será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista.

Comparación de terminología: Uno de los problemas que surge durante la licitación de requisitos es que usuarios y expertos no llegan a entenderse debido a problemas de terminología. Esta técnica es utilizada en forma complementaria a otras técnicas para obtener consenso respecto de la terminología a ser usada en el proyecto de desarrollo. Para ello es necesario identificar el uso de términos diferentes para los mismos conceptos *(correspondencia)*, una misma terminología para diferentes conceptos *(conflictos)* o cuando no hay concordancia exacta ni en el vocabulario ni en los conceptos *(contraste)*. [47]

Ingeniería Inversa: su objetivo es obtener información técnica a partir de un producto accesible al público, con el fin de determinar de qué está hecho, qué lo hace funcionar y cómo fue fabricado.

Es denominada ingeniería inversa porque avanza en dirección opuesta a las tareas habituales de ingeniería, que consisten en utilizar datos técnicos para elaborar un producto determinado. En general si el producto u otro material que fue sometido a la ingeniería inversa fueron obtenidos en forma apropiada, entonces el proceso es legítimo y legal. [48]

Aplicar ingeniería inversa a algo supone profundizar en el estudio de su funcionamiento, hasta el punto de de llegar a entender, modificar, y mejorar dicho modo de funcionamiento.

Áreas de la Ingeniería Inversa:

Redocumentación: es la creación o revisión de una representación equivalente semánticamente dentro del mismo nivel de abstracción relativo.

Recuperación de diseño: es un subconjunto de la ingeniería inversa, en el cual, aparte de las observaciones del sistema, se añaden conocimientos sobre su dominio de aplicación, información externa, y procesos deductivos con el objeto de identificar abstracciones significativas a un mayor nivel.

Rediseño: consiste en consolidar y modificar los modelos obtenidos, añadiendo nuevas funciones requeridas por los usuarios.

Reingeniería de Software: es el examen y alteración de un sistema para reconstruirlo de una nueva forma y la subsiguiente implementación de esta nueva forma. [49]

Descripción de la estrategia de la gestión de requisitos.

Para la Gestión de Requisitos del sistema primeramente se han considerado las técnicas de recopilación de la información, a través de reuniones y entrevistas con los usuarios/desarrolladores que utilizan herramientas para el desarrollo de multimedia, además de directivos de diferentes áreas de trabajo vinculados tanto al desarrollo de productos educativos como especialistas en otras materias. Se han revisado algunas descripciones de sistemas parecidos, documentos internos, manuales de procedimientos, leyes, normas y recomendaciones, algunos blog de foros en Internet acerca del tema.

Para desarrollar una colección de requisitos del sistema se ha seguido un conjunto de procesos, donde interaccionan los usuarios, el entorno y la comunidad técnica, ¿De qué manera?, pues los usuarios/desarrolladores de multimedia pueden aportar sus requisitos a nivel de descripción de datos, funciones y comportamientos, siendo estos la partida para que la comunidad técnica actúe como integrador, como consultor y negociador, y existe evidentemente un ciclo ininterrumpido realimentando la colección de requisitos, teniendo en cuenta el entorno en el cual se desarrolla que aporta sus restricciones para el análisis del sistema.

Los objetivos se evidencian claramente en el prototipo no funcional del sistema propuesto ya que se puede determinar la funcionalidad y capacidad del mismo, detectar interfaces externas, determinar el nivel de calidad, además sirven para fijar normas de seguridad, definir conjuntos de datos y relaciones, definir entornos de operación y ejecución.

La Gestión de Requisitos *(Funcionales y No Funcionales)* realizada para el IDE debe cubrir el sistema para que se desarrolle en gran medida la arquitectura, donde se observa si se está usando el RUP como metodología de desarrollo, y todo el análisis del problema de dominio y negocio que cubre la arquitectura.

3.2.2 Definición de requisitos.

Para la actividad de definición de requisitos en el proceso de ingeniería de requisitos existe un gran número de técnicas propuestas. Se describen a continuación las más relevantes:

Lenguaje natural: Resulta una técnica muy ambigua para la definición de los requisitos. Consiste en definir los requisitos en lenguaje natural sin usar reglas para ello. A pesar de que son muchos los trabajos que critican su uso, es cierto que a nivel práctico se sigue utilizando.

Glosario y ontologías: La diversidad de personas que forman parte de un proyecto de software hace que sea necesario establecer un marco de terminología común. Por esta razón son muchas las propuestas que abogan por desarrollar un glosario de términos en el que se recogen y definen los conceptos más relevantes y críticos para el sistema.

En esta línea se encuentra también el uso de ontologías, en las que no sólo aparecen los términos, sino también las relaciones entre ellos.

Plantillas o patrones: Esta técnica, recomendada por varios autores, tiene por objetivo el describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea ésta, menos ambigüedad ofrece. Sin embargo, si el nivel de detalle elegido es demasiado estructurado, el trabajo de rellenar las plantillas y mantenerlas, puede ser demasiado tedioso.

Escenarios: La técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos. La representación del escenario puede variar dependiendo del autor. Esta representación puede ser casi textual o ir encaminada hacia una representación gráfica en forma de diagramas de flujo. El análisis de los escenarios, hechos de una forma u otra, pueden ofrecer información importante sobre las necesidades funcionales del sistema.

Casos de uso: Como técnica de definición de requisitos es como más ampliamente han sido aceptados los casos de uso. Actualmente se ha propuesto como técnica básica del proceso RUP. Sin embargo, son varios los autores que defienden que pueden resultar ambiguos a la hora de definir los requisitos, por lo que hay propuestas que los acompañan de descripciones basadas en plantillas o de diccionarios de datos que eliminen su ambigüedad.

Lenguajes Formales: Otro grupo de técnicas que merece la pena resaltar como extremo opuesto al lenguaje natural, es la utilización de lenguajes formales para describir los requisitos de un sistema. Las especificaciones algebraicas como ejemplo de técnicas de descripción formal, han sido aplicadas en el mundo de la ingeniería de requisitos desde hace años. Sin embargo, resultan muy complejas en su utilización y para ser entendidas por el cliente. El mayor inconveniente es que no favorecen la comunicación entre cliente y analista. Por el contrario, es la representación menos ambigua de los requisitos y la que más se presta a técnicas de verificación automatizadas. [49]

Mediante el análisis de las técnicas existentes para definir los requisitos, se decidió utilizar el glosario, las plantillas, los escenarios y casos de uso.

El Glosario, para conceptualizar las terminologías y lograr un entendimiento entre equipo de desarrollo y el cliente.

Las plantillas, para estructurar de una forma estándar la información, previendo que el nivel de detalles no fuera tan estructurado evitando así las ambigüedades. Las plantillas seleccionadas fueron las que propone la metodología RUP.

Los casos de uso, se tomaron por ser considerados una técnica básica del proceso de RUP utilizado en el desarrollo del sistema. Éstos fueron descritos en plantillas estándares de RUP.

3.2.3 Validación de requisitos.

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias.

Aún así, existen algunas técnicas que pueden aplicarse para ello:

Reviews (Revisión): Está técnica consiste en la lectura y corrección completa de la documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida. Más difícil es verificar consistencia de la documentación o información faltante.

Auditorías: La revisión de la documentación con esta técnica consiste en un chequeo de los resultados contra una lista de chequeo predefinida o definida a comienzos del proceso, es decir sólo una muestra es revisada.

Matrices de trazabilidad: Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos.

Prototipos: Algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final. [49]

Para la validación de los requisitos se emplearon las técnicas: revisión y prototipos.

Posterior a la obtención de varios requisitos y del modelado de ellos se sostuvieron revisiones para cerciorarse que la interpretación por parte del analista fue correcta y que no faltó información.

Para obtener una idea de la interfaz de usuario se desarrolló un prototipo con parte de las funcionales críticas del sistema.

3.3 Requerimientos funcionales. (RF)

Una vez conocidos los conceptos que rodean al objeto de estudio, se hace necesario empezar a analizar ¿Qué debe hacer el sistema para que se cumplan los objetivos planteados al inicio de este trabajo?, para ello se enumeran a través de requerimientos funcionales las funciones que el sistema deberá ser capaz de realizar. Dentro de ellos se incluyen las acciones que podrán ser ejecutadas por el usuario, las acciones ocultas que debe realizar el sistema, y las condiciones extremas a determinar por el sistema. De acuerdo con los objetivos planteados el sistema debe ser capaz de:

R1. Manejar Contenido Multimedia (Recursos Multimedia).

1.1 Copiar objeto.

1.2 Cortar objeto.

1.3 Pegar objeto.

1.4 Eliminar objeto.

1.5 Duplicar objeto.

1.6 Seleccionar todos los objetos.

1.7 Modificar objeto.

1.8 Importar contenido multimedia.

1.8.1 Importar recursos multimedia (imágenes, gráficos, sonidos, video, etc).

1.8.2 Importar script.

R2. Gestionar Composición (Proyecto).

2.1 Crear composición.

2.2 Abrir una composición.

2.3 Guardar composición.

2.3.1 Guardar una composición.

2.3.2 Guardar parte de una composición.

2.4 Cerrar una composición.

2.5 Exportar composición

2.5.1 Exportar una composición.

2.5.2 Exportar parte de la composición.

R3. Controlar contenido de la Composición (Línea de Tiempo).

3.1 Crear capa.

3.1.1 Mostrar propiedades de la capa en otra ventana.

3.2 Crear carpetas de capas.

3.3 Insertar fotograma clave.

3.4 Copiar fotograma clave.

3.5 Pegar fotograma clave.

3.6 Eliminar fotograma clave.

3.7 Convertir fotograma en fotograma clave.

3.8 Manejar tiempo de los fotogramas.

3.9 Seleccionar todos los fotogramas.

R4. Manejar contenido interactivo.

4.1 Crear símbolo.

4.1.1 Crear símbolo Moviclip.

4.1.2 Crear símbolo Botón.

4.1.3 Crear símbolo Gráfico.

4.2 Insertar/ arrastrar componentes (objetos inteligentes) en la escena.

R5. Manejar flujo de animación.

5.1 Agregar efecto al objeto.

R6. Gestionar Herramientas.

6.1 Hacer figuras (ovalo, rectángulo).

6.2 Pintar figuras (aplicar relleno).

6.2.1 Mezclar colores.

6.2.2 Modificar relleno y color de borde.

6.3 Modificar ilustraciones.

6.4 Crear líneas rectas en algún lugar de la escena.

6.5 Crear texto en algún lugar de la escena.

6.5 Borrar elementos que se deseen.

R6. Mostrar ayuda genérica.

3.4 Requerimientos no funcionales. (RNF)

En este epígrafe se especifica los requerimientos no funcionales que se tuvieron en cuenta para el diseño y desarrollo de los prototipos.

Los requerimientos no funcionales son propiedades o cualidades que el producto debe cumplir. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable.

3.4.1 Requerimiento de Rendimiento y Disponibilidad del Sistema.

Se debe garantizar restablecimiento general del sistema durante falla de procesos, en menos de tres (3) segundos.

Se debe garantizar el restablecimiento del sistema en condiciones normales desde un estado desenergizado a un estado de completo funcionamiento en un período de cinco (5) minutos.

Se debe garantizar el apagado del sistema en condiciones normales desde su estado de operación normal a un estado de totalmente apagado en un período de cinco (5) minutos.

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.

3.4.2 Requerimiento de Soporte.

Se debe garantizar la realización de cursos auxiliares, preparatorios y de asesoría a los desarrolladores y usuarios del sistema.

Se debe garantizar mantenimiento y ampliación del sistema, al igual que las pruebas correspondientes para su correcto funcionamiento.

3.4.3 Requerimiento de Portabilidad.

Se debe garantizar la compatibilidad del sistema para múltiples plataformas, la familia de los SO GNU/Linux, Windows, Mac OS.

Se debe garantizar que un proyecto creado bajo Linux debe ser transparente bajo Windows.

Se debe garantizar compatibilidad en versiones, donde una composición creada bajo la versión 1 debe poder abrirse transparentemente bajo la versión 2 y viceversa con advertencias.

3.4.4 Requerimiento de Confiabilidad.

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

Se debe garantizar que el uso del sistema una vez instalado no requiera de permisos administrativos para su correcto funcionamiento.

Se debe garantizar que cada acción importante cuente con advertencias si la operación es incorrecta.

3.4.5 Requerimiento de Documentación de Usuario y Sistema de Ayuda en Línea.

Se debe garantizar que el IDE posea una ayuda que brinde al usuario en dependencia de la acción u objetos seleccionado una guía para trabajar, así como una breve descripción de cada vínculo activo en la interfaz gráfica para el desarrollo de las acciones habilitadas según el o los elementos seleccionados.

3.4.6 Requerimiento de Apariencia o interfaz externa.

Se debe garantizar que la interfaz de la aplicación sea lo más atractiva y clara posible para el usuario final.

Se debe garantizar que la interfaz sea amigable y de fácil comprensión en su funcionamiento permitiendo la utilización del sistema sin mucho entrenamiento, donde las estructuras visibles existentes tengan un conjunto de objetos visibles capaces de ser controlados por el usuario con iconos y botones.

3.4.7 Requerimiento asociados al Licenciamiento.

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

3.4.8 Requerimiento de Interfaces de Usuario.

(I) En el orden de prioridades de la implementación de los requisitos éste significa Inmediato.

(M) En el orden de prioridades de la implementación de los requisitos éste significa Mediano Plazo.

(L) En el orden de prioridades de la implementación de los requisitos éste significa Largo Plazo.

- La aplicación debe proveer al usuario una gran versatilidad en la presentación de la información en las pantallas, en cuanto a disponibilidad de los recursos y organización de los mismos. **(I)**
- El área de trabajo de despliegue propiciada, debe poderse personalizar.
- Debe permitir el manejo de ambientes multi-ventanas o despliegues, visualizándose varios despliegues simultáneamente, máximo 10 (configurable por usuario), sin generar efectos negativos en el rendimiento del sistema.
- 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. **(I)**
- 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.
- En la línea de tiempo de debe mostrar una capa y un fotograma clave por defecto, así como el escenario y las ventanas comunes.
- La aplicación debe estar sectorizada en varias áreas funcionales. **(I)**

- Para el Menú y las Barras de Herramientas, utilizar la parte superior.
- Para el editor de script y mensajes del sistema y errores utilizar la parte inferior.
- 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 mostrar opciones estándares (copiar, cortar, pegar, etc), pudiendo acceder mediante el clic derecho sobre un componente o por la combinación de teclas calientes. **(I)**
- Permitir mostrar opciones funcionales (agregar, modificar, eliminar, etc), pudiendo acceder mediante el clic derecho sobre un componente o por la combinación de teclas calientes. **(I)**
- Permitir tener un Menú que posea una serie de opciones que garanticen realizar acciones para el manejo y control de los objetos gráficos.
  - Mostrar Información en los laterales de la ventana principal.
  - Mostrar ventana de propiedades de cada recurso y componente.

3.5 Actores y casos de uso del sistema.

En este epígrafe se realiza el proceso de modelación del sistema para la realización de productos educativos multimedia. Para ello se identifican los actores y los casos de usos correspondientes para el funcionamiento.

3.5.1 Actor del Sistema.

Los actores representan personas o sistemas externos que interactúan con el sistema; actualmente para el IDE se considera un solo perfil de usuario:

Usuario: será cualquier persona vinculada o no a la producción de multimedia, tendrá el dominio y control de todas las funcionalidades del sistema para el desarrollo del producto.

3.5.2 Modelado del Sistema. Diagrama de Casos de Uso Críticos del Sistema.

Luego de haber definido los actores y requerimientos funcionales del sistema, se modela la relación que existe entre el actor Desarrollador de multimedia con los casos de usos críticos definidos a partir de la captura de los requerimientos.

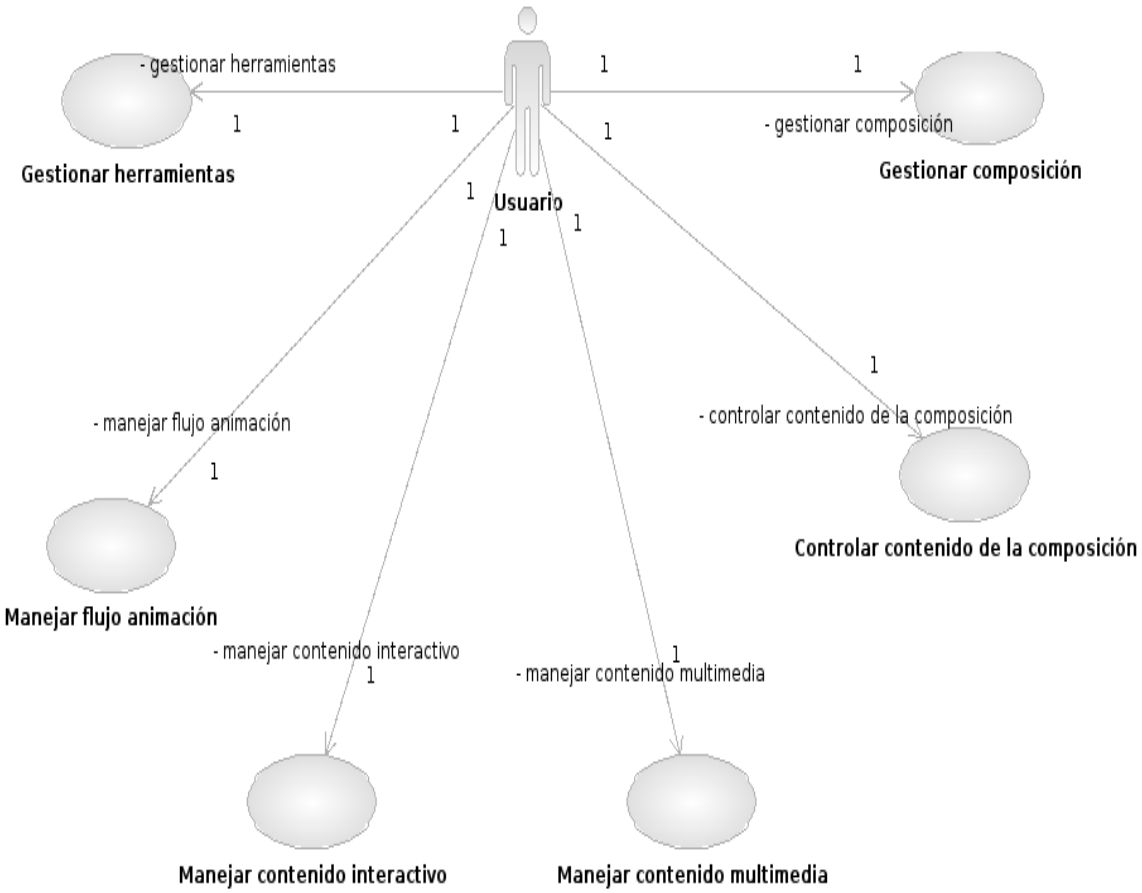

Figura 5. Diagrama de CU más significativos.

3.6 Conclusiones.

Para describir una propuesta de la solución se consideraron primeramente los antecedentes del tema, otros proyectos existentes, de esta manera se han planteado conceptos que aportan mucha claridad en el problema de dominio, así se ha construido el modelo de dominio y se han realizado buenas consideraciones para el entorno del IDE.

Además haciendo uso de las técnicas para la gestión de requisitos se obtuvo el levantamiento de requisitos del sistema, dándole realización a una parte de las actividades correspondiente a la etapa de análisis, se modeló el sistema y se concluyó con la validación de algunos requerimientos críticos mediante un prototipo no funcional.

Entre las técnicas escogidas para la captura de requisitos se encuentran el uso de las entrevistas, tormenta de Ideas, ingeniería inversa y comparación de terminología. Para la definición de los requisitos se decidió utilizar el glosario, las plantillas, los escenarios y casos de uso, posibilitando su validación a través de las técnicas: revisión y prototipos.

CONCLUSIONES

En la actualidad la situación socioeconómica está provocando fuertes vaivenes en el sector de las grandes compañías informáticas, lo que hace coger impulso a la ya concebida discusión entre software libre y comercial.

Por esas razones este trabajo aporta criterios para impulsar el proceso de migración que la dirección de software educativo de la UCI está implementando, ya que al existir un sistema para el desarrollo de productos educativos en formato multimedia se puede migrar totalmente sin problema alguno, porque se lograría una total independencia tecnológica, y se podría disminuir notablemente grandes gastos por las insostenibles licencias de software propietarios, también se lograría extender nuestros productos propiciando nuevas posibilidades para convenios con otros países.

- Para realizar el análisis del diseño del IDE primeramente se estudiaron las herramientas sobre la que se apoya la propuesta, herramientas para la construcción de la interfaz, herramientas para integrar y cumplir con los requerimientos que se requieren para el desarrollo de multimedia y generar el producto en el formato deseado.
- Se desarrolló la arquitectura del dominio introduciendo los conceptos más importantes para diseñar y proporcionar la aplicación global sobre su análisis, y estos se guardaron como definiciones en un glosario de términos y así el equipo de desarrollo tendría un vocabulario común.
- Se describió una propuesta de la solución considerando las técnicas de Gestión de Requerimientos, dándole realización a una parte de las actividades correspondientes a la etapa de análisis, se modeló el sistema con los casos de uso críticos y se concluyó con la validación de algunos requerimientos más significativos.
- Para realizar el levantamiento de requerimientos del sistema ha sido importante la participación constante de los especialistas de software educativos, los usuarios avanzados de GNU/Linux, para evitar la falta de identificación de los verdaderos requerimientos.
- La elaboración del prototipo de interfaz para el IDE, facilitará la comprensión de los casos de uso descritos.

RECOMENDACIONES

- Continuar el trabajo basándose en los requerimientos capturados no validados.
- Profundizar el estudio de nuevas alternativas de Software Libre para el campo de acción al cual está planteado el problema.
- Profundizar en el estudio de los reproductores de multimedia.
- Analizar el sistema de plugin necesario para que el IDE pueda integrar las herramientas propuestas.
- Usar los resultados de la captura de requerimientos para el diseño del IDE.

REFERENCIAS BIBLIOGRÁFICAS

[1] LINUXGLOBAL Proyecto GNU/Linux, 2007. http://www.linuxglobal.org/linuxglobal/gnu/

[2] WIKIPEDIA. Free Software Foundation, 2007. http://es.wikipedia.org/wiki/Free_Software_Foundation

[3] TECNOBLOGY. Distribuciones GNU/Linux, 2007. http://tecnoblogy.wordpress.com/todo-sobre-linux/

[4] STALLMAN., R. Software Libre para una sociedad libre, 2007: p.45. http://biblioweb.sindominio.net/pensamiento/softlibre/softlibre.pdf

[5] CONSORTIUM, W. W. Estándares y formatos libres, 2007. http://www.w3.org

[6] WIKIPEDIA. Licencia Open Source, 2007. http://es.wikipedia.org/wiki/Licencia_Open_Source

[7] WIKIPEDIA. Software Libre, 2007. http://es.wikipedia.org/wiki/Software_libre

[8] WKIPEDIA. XML, 2007. http://es.wikipedia.org/wiki/XML

[9] SANGUINETTI, C. Análisis y Diseño de Sistema.

[10] RUMBAUGH, J. and I. JACOBSON. El proceso unificado de desarrollo de software. La Habana, Addison Wesley Longman, 2000. p.

[11] TORRES, L. S. El analista de sistemas y el paradigma estructurado. www.monografias.com

[12] PRESSMAN, R. S. Ingeniería de software. Un enfoque práctico. Mc Graw Hill. 2005. p. 5a ed

[13] UNEX. Usos educativos de la informática. Herramientas de autor, 2007. http://www.unex.es/didactica/Tecnologia_Educativa/info03J.htm

[14] WIKIPEDIA. Flash For Linux (F4L), 2007. http://es.wikipedia.org/wiki/F4L

[15] WIKIPEDIA. QFlash, 2007. http://es.wikipedia.org/wiki/Qflash

[16] WIKIPEDIA. Uira, 2007. http://es.wikipedia.org/wiki/UIRA

[17] WIKIPEDIA. Ktoon, 2007. http://es.wikipedia.org/wiki/KToon

[18] KTOON Herramienta de Animación en 2D, 2007.

http://ktoones.toonka.com/index.php?option=com_content&task=view&id=5&Itemid=26

[19] WIKIPEDIA. TIC en la Educación, 2007.
http://es.wikipedia.org/wiki/Tecnolog%C3%ADas_de_la_informaci%C3%B3n_y_la_comunicaci%C3%B3n

[20] CORRALES, C. Una Nueva Tecnología de Comunicación e Información., 1994. http://iteso.mx/~carlosc/pagina/documentos/multidef.htm

[21] VILLANUEVA, A. B.
http://www.lcc.uma.es/~pastrana/EP/trabajos/02.pdf

[22] NICHO. Lenguajes Orientado a Objetos - C++, 2002.
http://www.ciao.es/Lenguajes_Orientado_a_Objetos_C__Opinion_524009

[23] WIKIBOOKS Programación con QT4, 2007.
http://es.wikibooks.org/wiki/Programaci%C3%B3n:Qt4

[24] WIKIPEDIA. Trolltech, 2007. http://es.wikipedia.org/wiki/Trolltech

[25] TROLLTECH. Novedades en Qt4.2, 2007.

http://doc.trolltech.com/4.2/qt4-2-intro.html

[26] TROLLTECH. The Arthur Paint System, 2007.
http://doc.trolltech.com/4.0/qt4-arthur.html

[27] WIKIPEDIA. KDevelop, 2007. http://es.wikipedia.org/wiki/KDevelop

[28] Compilador MTASC, 2006. http://www.mtasc.org

[29] OSFLASH SWFmill, 2007. http://osflash.org/swfmill

[30] FISCHER, D. SWFmill, 2007. http://www.swfmill.org/

[31] HAXE Lenguaje universal Haxe, 2007.
http://www.haxe.org/intro#introduction

[32] Swftoolls, 2007. http://www.swftools.org/documentation.html

[33] FERNANDEZ, J. M. Formatos Gráficos.
http://www.hsa.es/id/investigacion/uai/uai_docs/informatica/graficos.pdf

[34] WIKIPEDIA. Scalable Vector Graphics, 2007.
http://es.wikipedia.org/wiki/Scalable_Vector_Graphics

[35] WIKIPEDIA. Graphics Interchange Format (GIF), 2007.
http://es.wikipedia.org/wiki/Graphics_Interchange_Format

[36] WIKIPEDIA. Multiple image Network Graphics (MNG), 2007.
http://es.wikipedia.org/wiki/MNG

[37] LYCOS Formato MP3, 2007.
http://webmaster.lycos.es/topics/multimedia/music/music-introduction3/

[38] WIKIPEDIA. WAV, 2007. http://es.wikipedia.org/wiki/WAV

[39] WIKIPEDIA. Kubuntu, 2007. http://es.wikipedia.org/wiki/Kubuntu

[40] Ubuntu: GNU/Linux para gente normal, 2007.

http://www.ubuntu-es.org/ubuntu/introduccion

[41] WIKIPEDIA. Ubuntu (distribución Linux), 2007.
http://es.wikipedia.org/wiki/Ubuntu_(distribuci%C3%B3n_Linux)

[42] STALLMAN, R. La definición de software libre, 2007.
http://www.gnu.org/philosophy/free-sw.es.html

[43] STALLMAN, R. Copyleft, 2007.
http://www.gnu.org/copyleft/copyleft.es.html

[44] STREET, F. GNU Lesser General Public License, 2007.
http://www.gnu.org/copyleft/lesser.es.html

[45] GARCIA, R. R. and J. S. QUIEROS. Análisis de tendencias y estándares para la generación de gráficos vectoriales, 2007.

[46] ECLIPSE., F. D. Eclipse (software), 2007.
http://es.wikipedia.org/wiki/Eclipse_(computaci%C3%B3n)

[47] KOCH, N. Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo, 2007. www.pst.informatik.uni-muenchen.de

[48] WIKIPEDIA. Ingeniería Inversa, 2007. http://es.wikipedia.org/wiki/Ingenier%C3%ADa_inversa

[49] ALARCOS, G. Ingeniería Inversa, 2007. http://alarcos.inf-cr.uclm.es

[50] ANACHE, I. A. M. J., Ed. UML 2.0 Marcando un hito en el desarrollo de software, 2007.

BIBLIOGRAFIAS UTILIZADA

Formato de almacenamiento http://es.wikipedia.org/wiki/Formato_de_almacenamiento

Ley Francesa DADVSI http://es.wikipedia.org/wiki/DADVSI

Patentes de Software http://es.wikipedia.org/wiki/Patente_de_software

Documentación de Uira http://www.uira.org/wiki/doku.php?id=uira_use

ANEXOS.

Anexo 1

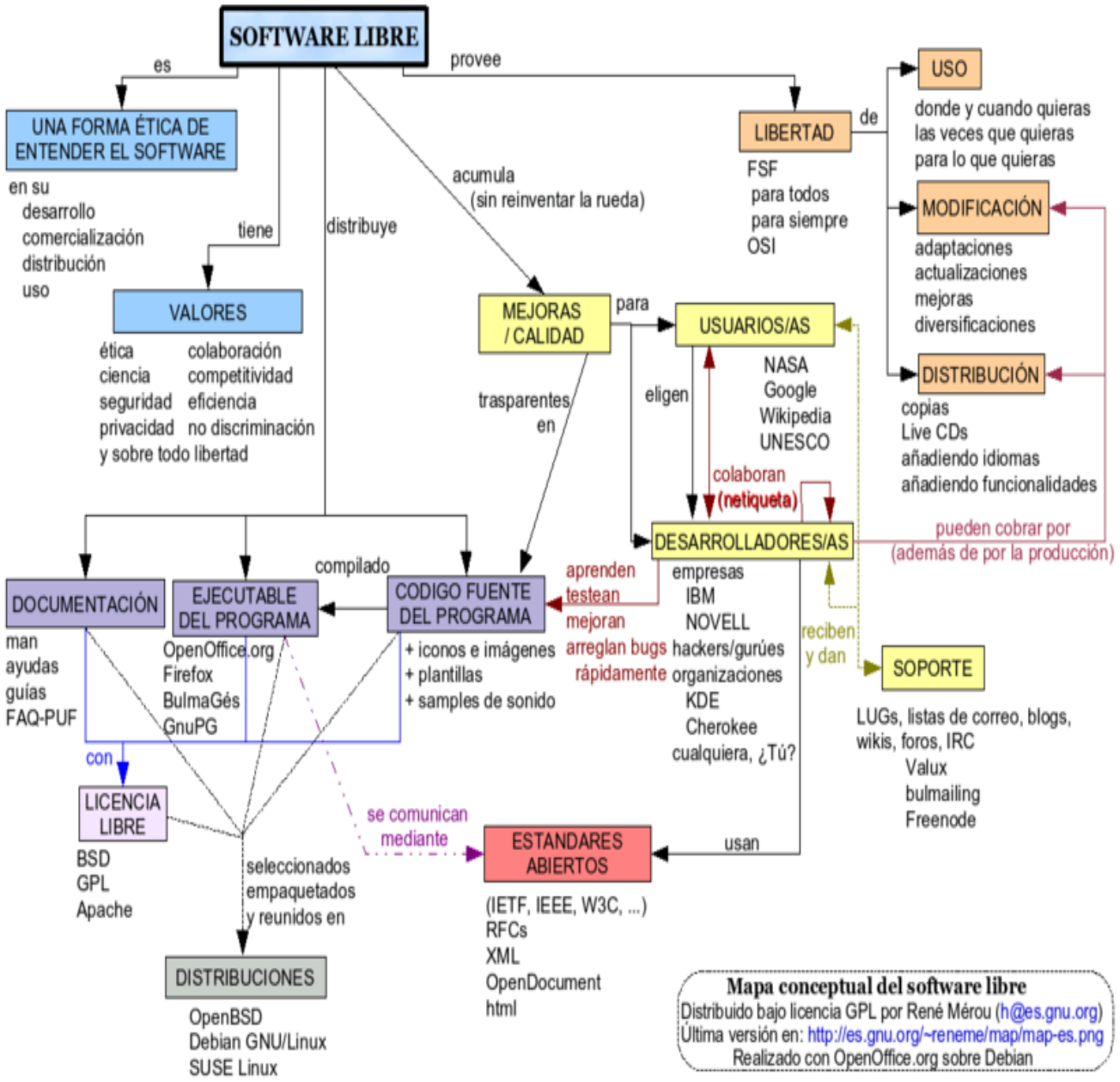

Figura 6. Mapa conceptual del software libre

Anexo 2

.frame 1

contenido frame

.end

Al insertar otro fotograma, pero en la posición 10 quedaría entonces:

.frame 1

contenido frame

.end

.frame 10

contenido frame

.end

GLOSARIO DE TÉRMINOS

API *(del inglés Application Programming Interface - Interfaz de Programación de Aplicaciones):* Es un conjunto de especificaciones de comunicación entre componentes software. Se trata del conjunto de llamadas al sistema que ofrecen acceso a los servicios del sistema desde los procesos y representa un método para conseguir abstracción en la programación, generalmente (aunque no necesariamente) entre los niveles o capas inferiores y los superiores del software. Uno de los principales propósitos de una API consiste en proporcionar un conjunto de funciones de uso general, por ejemplo, para dibujar ventanas o iconos en la pantalla. De esta forma, los programadores se benefician de las ventajas de la API haciendo uso de su funcionalidad, evitándose el trabajo de programar todo desde el principio. Las APIs asimismo son abstractas: el software que proporciona una cierta API generalmente es llamado la implementación de esa API.

BYTECODE es un código intermedio más abstracto que el código máquina. Habitualmente se le trata como a un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario que contiene código máquina producido por el compilador. Como código intermedio, se trata de una forma de salida utilizada por los implementadores de lenguajes para reducir la dependencia respecto del hardware específico y facilitar la interpretación.

CASE *(Computer Aided Software Engineering):* Es un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

CSCL: Es el Aprendizaje Cooperativo Soportado por Computadora *(en inglés Computer Supported Cooperative Learning)* , de innovación pedagógica para el desarrollo de software que está formando un nuevo modelo educativo basado en aumentar la interacción entre los miembros de un grupo cuyo interés común es el aprendizaje y que cuenta con el computador como el elemento para interactuar y el medio para comunicarse.

DCOP *(**D**esktop **CO**mmunication **P**rotocol, 'protocolo de comunicación de escritorio'):* Es un sistema de comunicación ligero entre procesos y componentes software. El principal fin de este sistema es permitir la interoperación de aplicaciones y compartir tareas complejas entre estas. Esencialmente, DCOP es un sistema de control remoto, que permite a una aplicación o script solicitar ayuda de otras aplicaciones.

DTD *(Document Type Definition)* es un conjunto de reglas para definir un documento XML y etiquetarlo adecuadamente. En una DTD definimos los "componentes" de un documento XML y cómo se relacionan y estructuran.

DOM *(Document Object Model):* Es una plataforma e interfaz neutral en lenguaje que permitirá programas y letras para dinámicamente ganar acceso y actualizar el contenido, estructura y el estilo de documentos. El documento puede ser adicionalmente tratado y los resultados de ese procesamiento pueden ser incorporados de vuelta a la página presentada.

FSF *(Free Software Foundation):* Es la Fundación para el Software Libre (FSF) está dedicada a eliminar las restricciones sobre la copia, redistribución, entendimiento, y modificación de programas de computadoras.

GNOME: GNU **N**etwork **O**bject **M**odel **E**nvironment es un entorno de escritorio basado en las bibliotecas GTK diseñadas para GIMP para sistemas operativos de tipo Unix bajo tecnología X Window. Al igual que KDE, permite la interacción entre programas. Se encuentra disponible actualmente en más de 35 idiomas. Forma parte oficial del proyecto GNU.

GUI: Graphical User Interface *(Interfaz gráfica de usuario)* es un método para facilitar la interacción del usuario con el ordenador a través de la utilización de un conjunto de imágenes y objetos pictóricos (iconos, ventanas) además de texto.

GCC *(Colección de compiladores GNU):* Es una colección de compiladores que admite varios lenguajes: C, C++, Objective C, Chill, Fortran y Java así como también las librerías para éstos. Surgió de la imposibilidad de compilar en un entorno que no fuese PC bajo MS-DOS.

GPL *(General Public License):* Es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

GNU (): Proyecto lanzado en 1984 para desarrollar un completo sistema operativo tipo Unix, bajo la filosofía del software libre: el sistema GNU. Las variantes del sistema operativo GNU que utilizan el núcleo llamado Linux.

GNU/Linux *(GNU con Linux o GNU+Linux)* es la denominación defendida por Richard Stallman y otros para el sistema operativo que utiliza el kernel Linux en conjunto con

las aplicaciones de sistema creadas por el proyecto GNU y de varios otros proyectos/grupos de software. Comúnmente este sistema operativo es denominado como Linux, aunque Stallman sostiene que esta denominación no es correcta.

GDK *(GIMP Dibujando a Kit):* Es una biblioteca de gráficos de la computadora que actúa como una envoltura alrededor del dibujo de bajo nivel y las funciones de recorte previstas por el sistema subyacente de gráficos. Originalmente desarrollado en el Sistema de Window. Manipula la representación gráfica en forma de trama (bitmaps), los cursores, los caracteres de imprenta, la funcionabilidad de arrastrar y colocar.

IDE (Integrated Development Environment): 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.

JDK *(Java Development Kit):* Es un compilador y conjunto de herramientas de desarrollo para la creación de programas independientes y applets java.

Kate es un editor de textos para KDE. Kate significa **K**DE **A**dvanced **T**ext **E**ditor, es decir Editor de textos avanzado para KDE.

KDE: K Desktop **E**nvironment es otro entorno de escritorio de gran popularidad, con gestor de ventanas propio y barra de tareas. Está programado en el lenguaje C++ y se basa en las bibliotecas QT+, lo que le ha significado ser víctima de muchas críticas por parte de la comunidad GNU/Linux, ya que estas bibliotecas eran propiedad de una empresa comercial.

LGPL *(Lesser General Public License GNU):* Es la Licencia Pública General Reducida de GNU la cual permite el enlazado de aplicaciones o bibliotecas libres en aplicaciones de software privativo, pero nunca permite que el código fuente de la aplicación liberada bajo GNU/LGPL sea propietaria o que fragmentos de dicho código sean incrustados en aplicaciones propietarias.

MTASC (**M**otion **T**win **A**ction **S**cript **C**ompiler): Es un compilador open source independiente de plataforma para generar swf, soporta ActionScript 2.0 y es capaz de compilar 100 clases en menos de 5 segundos.

Scripts: Conjunto de líneas de código que permite interactuar con los objetos del proyecto, llámense objetos gráficos, drivers, alarmas, etc, para dotar al sistema de

funcionalidades avanzadas que no pueden lograrse con recursos, eventos y comandos que propicia el sistema. Los scripts pueden ser de tres tipos: el primer tipo son los denominados Script Básicos, dado que pueden ser invocados desde cualquier contexto del proyecto, desde menús desde otros scripts, etc; los segundos son denominados scripts de objetos, estos scripts pueden ser ejecutados a partir de la ocurrencia de algún evento sobre objetos gráficos; a los últimos se les denomina Expresiones, los cuales no son más que ecuaciones que pueden ser evaluadas y que pueden contener, funciones matemáticas y variables de proceso asociadas, estos comúnmente son asignados a propiedades de los objetos que pueden ser animadas, que son en función de expresiones *(no sólo de variables).*

Widget *(Objeto gráfico):* Objeto gráfico que puede contener gráficos vectoriales o imágenes raster, (imágenes de mapas de bits) y presenta un conjunto de propiedades que pueden ser editadas y asociadas a puntos *(variables)* o a expresiones que contengan variables del sistema. Por medio de los objetos gráficos se pueden crear animaciones en función de los valores de los puntos asociados. Los widgets pueden agruparse para formar nuevos objetos gráficos más complejos. Un ejemplo de widget puede ser un contenedor de textos.*(MOYA 2004)*

OMG: Object Management Group *(Grupo de Gestión de Objetos).* Es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización NO lucrativa que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para tecnologías orientadas a objetos.

OS *(Open Sourse):* Es el término por el que se conoce el software distribuido y desarrollado libremente. Este término empezaron a utilizarlo en 1998 algunos usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original en inglés del software libre (*free software*).

OSFLASH: No es un proyecto para crear una alternativa Open Source FLASH IDE, sino una colección de herramientas y recursos open source para desarrolladores accesibles interesados en la plataforma Flash.

OSDN *(Open Source Development Network)*: Consorcio Global de una sociedad dedicada al auge del Software Libre.

POO *(Programación Orientada a Objetos):* La Programación Orientada a Objetos (POO u OOP según siglas en inglés) es un paradigma de programación que define los programas en términos de "clases de objetos", objetos que son entidades que

combinan *estado* (es decir, datos), *comportamiento* (esto es, procedimientos o *métodos*) e identidad (propiedad del objeto que lo diferencia del resto).

RUP: El Proceso Unificado de Rational (*RUP, el original inglés Rational Unified Process)* es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

SAX *(Simple API for XML):* Es una especificación ampliamente usada que describe cómo los analizadores gramaticales XML pueden pasar la información eficazmente de documentos XML a los programas de aplicación. SAX fue originalmente implementado en Java, pero es ahora soportado por casi todo principal lenguajes de programación.

SGML son las siglas de "Standard Generalized Markup Language" o "Lenguaje de Marcación Generalizado". Consiste en un sistema para la organización y etiquetado de documentos. La Organización Internacional de Estándares *(ISO)* ha normalizado este lenguaje en 1986.

SWF: Es un formato de archivo de gráficos vectoriales creado por el programa Flash de Adobe *(antes Macromedia).*

SWT *(Standard Widget Toolkit):* Creado por IBM como reemplazo de la Java Abstract Windowing Toolkit (AWT) y Swing. El SWT es un conjunto de widgets para desarrolladores Java que ofrece una API portable y una integración muy ligada con la interfaz gráfica de usuario nativa al sistema operativo de cada plataforma. Así, cada plataforma debe adaptar el SWT para los gráficos nativos suyos. Hasta el momento los entornos a los que se ha portado SWT son Windows, Linux GTK, Linux Motif, Solaris Motif, AIX Motif, HPUX Motif, Photon QNX, y Mac OS X.

Textfields: Es n campo de texto es un control básico que permite al usuario teclear una pequeña cantidad de texto y dispara un evento action cuando el usuario indique que la entrada de texto se ha completado.

TIC (Tecnologías de la Información y Comunicaciones) son las tecnologías que se necesitan para la gestión y transformación de la información, y muy en particular el uso de ordenadores y programas que permiten crear, modificar, almacenar, proteger y recuperar esa información.

UML *(Lenguaje Unificado de Modelado):* preescribe 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. Es un lenguaje

gráfico para visualizar, especificar, construir y documentar un sistema de software.

XSL (siglas de Extensible Stylesheet Language, expresión inglesa traducible como "lenguaje extensible de hojas de estilo") es una familia de lenguajes basados en el estándar XML que permite describir cómo la información contenida en un documento XML cualquiera debe ser transformada o formateada para su presentación en un medio específico. Es basado en XML y se diseñó para transformar un documento XML en cualquier otro documento estructurado.

W3C (*World Wide Web Consortium*): Es un consorcio internacional que produce estándares para la World Wide Web, fue fundada en octubre de 1994.