**UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS**

**Facultad 8**


**TRABAJO DE DIPLOMA PARA OPTAR**
**POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS**

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

**AUTORES:** Joan Pablo Jiménez Milian
Roberto Ferrer Obregón

**TUTOR:** Ing. Abel Ernesto Lorente Rodríguez
**COTUTOR:** Ing. Michel Miranda Cairo

**Ciudad de La Habana**
**Junio, 2008**

## DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo con carácter exclusivo.

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

**Autores:** Roberto Ferrer Obregón

Joan Pablo Jiménez Milian

**Tutor:** Ing. Abel Ernesto Lorente Rodríguez

_____

Firma de los autores

_____

Firma del tutor

## AGRADECIMIENTOS

***De Joan Pablo:***

*A la Revolución por la oportunidad de cumplir mis metas y sueños, y sobre todo al nuestro Comandante en Jefe Fidel por la guía y el ejemplo que nos transmite diariamente.*

*A mis padres y abuelos sin los cuales no habría podido lograrlo, por la educación y el apoyo que siempre me brindan.*

*A Ibis Baeza por su paciencia y apoyo durante toda esta jornada.*

*A Antonio Membrides que con su ayuda se ha convertido ya en parte del equipo de desarrollo.*

*A Roberto Ferrer mi colega, que su labor ha sido imprescindible para concretar este trabajo.*

***De Roberto:***

*A mi familia, por su apoyo y preocupación.*

*A mis padres, por su eterna confianza y por creer en mí cuando más lo necesitaba.*

*A mi hermana, por ser la luz de mis ojos y mi fuente constante de inspiración.*

*A Marlon y Janier por su fiel amistad, tan necesaria en los momentos difíciles.*

*A mis amigos, que de ponerlos todos sería una lista interminable.*

*A Joan Pablo, mi compañero de tesis, por su empeño en el desarrollo del software, y su espíritu optimista.*

***De manera general agradecemos:***

*A Abel Ernesto Lorente, nuestro tutor, por la confianza depositada en nosotros.*

*A Michel Miranda, por su asesoría e interés constante en lo que hacíamos.*

*A Henry Bermúdez por su colaboración desinteresada.*

*A todo el equipo de Primavera por apoyarnos y contribuir al proyecto.*

*Al resto del equipo de desarrollo: Nidia Fernández, Yonnys Pablo Martin, Yosber Rodríguez, Anisley de la Caridad y Geyser Zamora, por el excelente trabajo y ayuda que día a día nos ofrecieron.*

*A todos nuestros amigos y compañeros, que de una forma u otra aportaron su granito de arena para que este sueño se hiciera realidad.*

*Y en especial a la Revolución y a nuestro Comandante en Jefe Fidel.*

**DEDICATORIA**

*A nuestros padres, abuelos, y demás familiares, que nos apoyaron incondicionalmente y nos educaron en los principios socialistas y revolucionarios con los que crecimos.*

*A nuestro Comandante en Jefe por hacer posible nuestros sueños y ser el precursor de esta gran obra que es la Universidad de las Ciencias Informáticas.*

*A la Revolución, por todo lo que nos ha brindado.*

## RESUMEN

Actualmente el desarrollo de productos multimedia software educativo en la Universidad de las Ciencias Informáticas (UCI) y a nivel nacional, se realiza utilizando herramientas de autor propietarias, principalmente Adobe Flash y Macromedia Director, las cuales no cuentan con las licencias apropiadas para su explotación debido al bloqueo norteamericano que sufre Cuba desde 1960, haciendo casi imposible la comercialización de los mismos. A esto se une la progresiva migración a software libre que se lleva a cabo a nivel nacional, específicamente hacia GNU/Linux, que entra en contradicción con este tipo de herramientas totalmente privativas y diseñadas para ejecutarse sobre la plataforma Microsoft Windows, del cual, tampoco se cuenta con las licencias.

En este libro se hace un estudio profundo del estado del arte de las actuales herramientas existentes en el mundo del software libre para el desarrollo de software educativo con tecnología multimedia, además se propone e implementa un Ambiente de Desarrollo Integrado ( IDE ) que permita la migración de la producción de este tipo de productos a plataformas libres.
En los siguientes capítulos se abordarán estas temáticas hasta llegar a la solución propuesta.

**ÍNDICE**

## INTRODUCCIÓN

Existe en la actualidad un amplio consenso acerca del impacto que tienen las Tecnologías de la Información y las Comunicaciones (TICs) en varios niveles de la sociedad. Las TICs emergen como instrumentos novedosos, generándose una amplia gama de utilidades para éstas, siendo una de sus principales condiciones la de funcionar como eficaces herramientas para el desarrollo. Esta correlación entre TICs y desarrollo ha sido ampliamente estudiada e investigada y, por lo general, se considera que existe una asociación positiva entre ambas variables, de modo que la inversión en TICs se considera una dimensión importante en la consecución exitosa de proyectos de desarrollo. Las ventajas que ofrece el uso de las tecnologías de información y comunicación al mundo industrializado (comercio electrónico, aceleración en los procesos de decisiones, acceso a información en-línea, etcétera.), están ligadas sin duda a lo que los expertos han llamado 'creatividad económica de las naciones'. Investigaciones recientes han demostrado una fuerte conexión entre el uso de las TICs, el desarrollo de prácticas innovadoras, y el crecimiento económico (Choike 2007).

La educación no ha estado ajena al impacto de las nuevas tecnologías de la información y las comunicaciones. La palabra "multimedia" en Educación se utiliza desde mucho antes que fuera incorporado al léxico de los soportes comunicativos. Se hablaba de programas de enseñanza multimedia que utilizaban la radio, la televisión y la prensa para alfabetizar o enseñar idiomas. También los paquetes multimedia de uso didáctico incluían cintas de audio junto a materiales impresos y audiovisuales con contenidos instructivos de diferentes cursos a impartir, idioma, contabilidad, etc. En la actualidad con la introducción de la computadora el término multimedia adquiere otro significado, refiriéndose a aquellos programas que integran audio, vídeo, imágenes y texto para transmitir contenidos informativos o educativos.

Un software con tecnología multimedia puede considerarse **software educativo**, o **software educativo multimedia,** si ha sido diseñado para cumplir objetivos docentes y científicos, ya sea como apoyo a determinada materia o como medio de enseñanza de contenidos didácticos, existiendo una gran variedad de productos de este tipo. Actualmente el impacto de las multimedias educativas ha sido tan grande que existe una tendencia cada vez mayor de los gobiernos en utilizarlas como plataformas de aprendizaje virtual en la mayoría de los centros educacionales.
El software educativo es un concepto mucho más abarcador, es todo aquel software destinado a la enseñanza y el auto aprendizaje a través de la computadora, garantizando un nivel de interactividad que permite a los estudiantes desarrollar habilidades cognitivas.

Cuba se encuentra inmersa en una enorme Batalla de Ideas, por lo que es indispensable elevar al máximo el nivel escolar y cultural de la sociedad cubana. Como parte del desarrollo del mundo actual se ha hecho necesaria la modificación de los medios tradicionales de enseñanza, introduciéndose las TICs en Universidades y escuelas de todo el país. Una de las mayores transformaciones ha constituido la introducción de la computadora en las aulas, como material de apoyo a las asignaturas que se imparten, permitiendo a los estudiantes aprender de manera interactiva y didáctica a través de diversos programas educativos existentes, lo cual favorece el aprendizaje y desarrolla nuevas habilidades del conocimiento en el uso de las nuevas tecnologías.

Sin embargo, a pesar de que el software educativo se adopta cada vez más en la educación cubana, el principal problema sigue siendo lograr captar la atención del estudiante y motivarlo a que aprenda y utilice este tipo de software, para ello se hace necesario combinar adecuadamente los medios disponibles, vídeos, sonido, imágenes y textos, ya sea en un documento, programa o presentación, de esta forma se mejora notablemente la atención, la comprensión y el aprendizaje, acercándose algo más a la manera habitual en que los seres humanos se comunican, cuando se emplean varios sentidos para comprender un mismo objeto o concepto, obteniéndose como resultado productos multimedia educativos. La UCI juega un importante papel en la informatización de la sociedad cubana, así como en la producción de software, y particularmente en la producción de software educativo multimedia, como resultado existen numerosos proyectos encaminados a la creación de multimedia educativas.

Conjuntamente a esto, se está llevando a cabo una política de migración a software libre, cuyo objetivo principal es lograr la independencia tecnológica de los grandes monopolios de la industria del software capitalista. Otro factor importante para la migración es el hecho de que como consecuencia del injusto bloqueo que sufre el pueblo cubano desde hace más de 40 años, Cuba no cuenta con las licencias de software necesarias para la producción de cualquier tipo de aplicación, incluido multimedia, además de los costos de estas licencias que hacen poco sostenible adquirirlas.

En contraposición a lo anterior, en la UCI actualmente todo el software educativo con tecnología multimedia se desarrolla utilizando herramientas privativas, rompiendo así con las necesidades de migración a plataformas y tecnologías libres. Hoy día la escasez de herramientas libres que satisfagan las necesidades de producción de multimedia para plataformas libres es uno de los principales inconvenientes que está afectando la migración.

**Situación Problémica**

Cuba está llevando a cabo una estrategia de migración a software libre a nivel nacional, para lograr así la independencia tecnológica que permitirá alcanzar un mayor desarrollo en las nuevas tecnologías y el mundo actual. Por otra parte, como resultado de la implantación de las nuevas tecnologías en las escuelas y centros educacionales, existe una creciente demanda de software educativo, específicamente Multimedia, que sirvan como material de apoyo para el aprendizaje de los estudiantes.

La Universidad también se encuentra en un proceso de migración, comenzando por los proyectos productivos, sin embargo todo el software educativo y multimedia se desarrolla con herramientas de autor propietarias como Adobe Flash y Macromedia Director, rompiendo con la políticas de migración establecidas, además de que debido al bloqueo norteamericano no se cuenta con licencias para el uso de estas herramientas, lo cual hace imposible la exportación de estos productos.

Por otra parte, estas herramientas privativas orientadas al desarrollo de software multimedia están sólo destinadas a la plataforma también privativa Microsoft Windows. Esto trae como consecuencia la imposibilidad de utilizarlas en plataformas libres como GNU/Linux, independientemente de lo costoso y poco rentable de las licencias de “El Gigante Monopolio Microsoft”, atribuyendo además el bloqueo impuesto por EEUU a Cuba que prohíbe a Empresas Norteamericanas el comercio con la isla. En la actualidad la escasez de este tipo de herramientas en plataformas libres y con tecnologías libres es una realidad que afecta el proceso de migración.

**Problema científico**

La no existencia de herramientas que faciliten la creación y edición de código para el desarrollo de productos multimedia educativos sobre plataformas y tecnologías libres, y la falta de documentación y conocimientos sobre las pocas que existen y que en muchas ocasiones son insuficientes para el desarrollo de proyectos complejos, impiden cumplir con las necesidades de producción de software educativo multimedia en la Universidad de las Ciencias Informáticas.

**Objeto de Estudio**

Desarrollo de un Editor de Código que permita y facilite la gestión de código en el Ambiente de Desarrollo Integrado (IDE) propuesto.

**Campo de Acción**

Desarrollo de productos multimedia y software educativo en la UCI.

**Objetivos Generales**

Proporcionar una herramienta para la gestión de código en el Ambiente de Desarrollo Integrado (IDE) propuesto para la creación de aplicaciones de software educativo multimedia.

**Tareas**

1. Revisión Bibliográfica.
2. Actualización de la Documentación.
3. Definir situación problémica, problema científico, objeto de estudio, campo de acción, objetivo general.
4. Elaborar Cronograma de Actividades.
5. Análisis de las diferentes herramientas existentes de gestión de código para multimedia, fundamentalmente en el mundo del software libre.
6. Estudio de los posibles componentes a utilizar que permitan un conjunto de funcionalidades para la gestión de código.
7. Evaluar soluciones y herramientas encontradas.
8. Selección y fundamentación de una metodología de desarrollo
9. Levantamiento de Requisitos.
10. Revisión de Requisitos.
11. Definir Casos de Uso.
12. Detallar Casos de Uso.
13. Estructurar Modelo Casos de Uso.
14. Realizar un prototipo de interfaz de usuario.
15. Evaluación de los patrones de arquitectura.
16. Seleccionar un IDE de desarrollo.
17. Implementación de un prototipo no funcional.
18. Realizar el análisis y diseño.
19. Implementación de los Casos de Uso más importantes.
20. Documentar todo el trabajo realizado.
21. Estructurar el documento de tesis.

**Antecedentes**

La UCI al ser una universidad relativamente joven, no cuenta con experiencia en el desarrollo de una herramienta para el desarrollo de software educativo multimedia, por lo que se realizó un estudio profundo del estado del arte de aplicaciones libres de este tipo. Como resultado se encontró que varios proyectos han intentado desarrollar una herramienta alternativa a Macromedia Flash, ahora Adobe Flash, sin embargo, nunca fueron completamente implementados, quedando solo en la etapa de inicio.

Esta investigación hizo clara la necesidad de desarrollar un software que integre las actuales herramientas disponibles para desarrollar software multimedia, ajustándolo a las necesidades de la UCI y del país en general. Para facilitar el proceso de desarrollo, debido al nivel de complejidad de la aplicación que se desea implementar, se dividió el sistema en tres subsistemas:

- Subsistema de Gestión Visual.
- Subsistema de Gestión de Código.
- Subsistema Ejercicios y Clases Reutilizables.

Este libro está centrado en el desarrollo del Subsistema de Gestión de Código, asumiendo como objetivo fundamental obtener un producto estable que garantice la gestión de código en el IDE, así como documentar todo el proceso de análisis, diseño y desarrollo del mismo.

**Estructuración del contenido por capítulos**

**Capítulo 1:** Fundamentación Teórica. En este capítulo se plantean los elementos teóricos que sustentan el trabajo, se estudia el estado del arte y se describen las soluciones existentes. Además se hace un profundo análisis de las principales tendencias y tecnologías actuales, con el objetivo de seleccionar los componentes y herramientas a utilizar para la solución del problema planteado.

**Capítulo 2:** Análisis de la Solución Propuesta. En este capítulo se realiza el modelo del negocio, utilizando un Modelo de Dominio, debido a la poca estructuración de los procesos del negocio. Se hace el Levantamiento de Requisitos, tanto funcionales como no funcionales, se realiza el Modelo de Casos de Usos del Sistema y se describe cada uno de ellos.

**Capítulo 3:** Construcción de la solución propuesta. En este capítulo se construye la solución propuesta, se presentan los Diagramas de Clases de Análisis y Diseño, Diagrama de Componentes, Diagrama de Despliegue, y se obtiene como resultado una aplicación completamente funcional, que sirve de solución al problema planteado.

**Capítulo 4:** Análisis de Costos y Factibilidad. En este capítulo se presenta un estudio completo de factibilidad del producto, estimándose el tiempo de desarrollo que se requiere para la implementación del sistema, el esfuerzo humano y los costos económicos, así como los beneficios tangibles e intangibles que se obtendrán una vez desarrollado el producto.

# CAPÍTULO 1

## Fundamentación Teórica.

### 1.1 Introducción

En este capítulo se tratarán importantes aspectos relacionados con la fundamentación teórica, así como un análisis del estado del arte del problema. También se analizarán las soluciones actuales y se describirán cada una de ellas, así como las licencias de software. Es objetivo de este capítulo hacer una descripción general del Objeto de Estudio e identificar la audiencia a quién va dirigida la propuesta. Por último se explican varios conceptos asociados con el dominio del problema.

### 1.2 Análisis de otras soluciones existentes

Actualmente las herramientas de autor para la producción de software educativo multimedia se encuentran centradas en la plataforma Microsoft Windows, además de que constituyen software privativo, con una serie de restricciones que nos impiden su utilización legal y económica. Sin embargo, existen un conjunto de herramientas libres desarrolladas por la comunidad que brindan alternativas parciales a los estos programas, posibilitando el proceso de migración a software libre. A continuación se describen cada una de ellas.

**1.2.1 F4L:** Fue uno de los primeros proyectos orientados a sustituir Macromedia Flash. Ofrecía una interfaz de usuario similar a la de Flash MX, y permitía el dibujo y animación de elementos básicos, sin embargo poseía problemas de diseño a nivel de arquitectura del software, por lo que fue abandonado por los desarrolladores. (f4l Team 2005)

**1.2.2 Qflash:** Constituyó la nueva alternativa para desarrollar un clon del Adobe Flash, objetivo que lograron en el diseño de la interfaz. Entre los principales logros de este proyecto destacan: dibujo de formas básicas, creación de animaciones fotograma a fotograma, trabajo con capas, exportar a SWF. Este proyecto fue abandonado para crear UIRA (Sergi Pérez Maña 2007).

**1.2.3 UIRA:** Pretendía ofrecer una completa alternativa al Adobe Flash, pensando desde los inicios en una arquitectura flexible que evitara los problemas de los dos proyectos anteriores. Nace de la unión de los programadores de f4l y Qflash. Este proyecto generó muchas expectativas, sin embargo no llegaron a desarrollar un prototipo funcional que brindara una solución inicial. Actualmente es posible obtener el código fuente del prototipo no funcional (Florian Delizy 2007).

**1.2.4 Ktoon:** KToon es una herramienta para desarrollo de animaciones en 2D, diseñada por animadores para animadores, con un enfoque a la industria de la animación profesional. Este proyecto está cubierto por la Licencia GPL, y es producido por una empresa colombiana llamada Tomka Films. Entre las grandes posibilidades que ofrece para el dibujo y el trabajo con capas, destaca su posibilidad de exportar a varios formatos, como por ejemplo SWF. Este proyecto no pretende ser una copia de Flash y no permite la creación de contenido interactivo (Tomka Films 2008).

**1.2.5 Flash Develop:** Es un completo editor de código para ActionScript, admitiendo hasta la versión 3 de este lenguaje. Es desarrollado usando el .NET Framework de Microsoft, que limita su uso solo a sistemas operativos de la familia Microsoft Windows. Utiliza como base para generar los archivos SWF el SWFMill y el MTASC (Mika Palmu 2008).

**1.2.6 SEPY:** SE|PY ActionScript Editor. Es un editor para ActionScript 2.0 orientado principalmente a plataformas Microsoft Windows, desarrollado en python. Incluye completamiento básico de código, resaltado de sintaxis, explorador de clases e integración con proyectos de Adobe Flash, entre otros, sin embargo desde hace dos años no cuenta con ninguna actualización, por lo que ha quedado obsoleto frente a las nuevas tecnologías y herramientas de desarrollo que surgen (Alessandro Crugnola 2006).

**1.2.7 MTASC:** Motion Tween Action Script Compiler. Es un compilador libre para ActionScript. Soporta hasta la versión 2.0. Exige que el código que se escriba sea completamente orientado a objetos y es mucho más veloz que el compilador interno de Adobe Flash. No posee interfaz gráfica de usuario (GUI) que facilite su manejo, siendo una herramienta modo consola, dificultad que hace que algunos usuarios no lo utilicen por temor a complicarse. A pesar de que MTASC se encuentra en una versión estable, se ha dejado de desarrollar para dar paso a un nuevo compilador llamado haXe (Motion Tween 2007).

**1.2.8 SWFMILL:** Es un compilador modo consola basado en los procesadores XLST que convierte archivos XML en SWF, estos archivos se escriben usando SWFML, un lenguaje XML creado específicamente para ser similar al formato SWF. Este lenguaje tiene etiquetas propias que permiten

definir propiedades de la película final y crear objetos dentro de ella, así como importar imágenes y sonido. Generalmente se utiliza conjuntamente con el MTASC (Daniel Fischer 2008).

**1.2.9 haXe:** Este compilador ha sido desarrollado por los mismos programadores que hicieron el MTASC. Entre las características fundamentales que posee destacan:

- Soporte para ActionScript 1,2,3.
- Soporte para JavaScript.
- Posee además un lenguaje propio (haxe).
- Posee una máquina virtual llamada neko que permite agregar funcionalidades como escritura en archivos.
- No posee interfaz visual. (Motion Tween 2008).

**1.2.10 SWFTools:** Son un conjunto de herramientas libres que permiten el manejo y creación de archivos SWF. Está compuesto por:

- **PDF2SWF**: Convierte archivos PDF a SWF. Genera por cada página existente en el documento un fotograma dentro de la película flash.
- **SWFCombine:** Inserta SWFs en archivos SWF contenedores. Como su nombre lo indica, permite combinar dos SWF para formar uno.
- **SWFStrings:** Analiza el archivo SWF en busca de Texto.
- **SWFDump:** Imprime información sobre determinado SWF.
- **JPEG2SWF:** Permite generar una presentación en SWF a partir de una o varias imágenes JPEG.
- **PNG2SWF:** Permite generar una presentación en SWF a partir de una o varias imágenes PNG.
- **GIF2SWF** Convierte archivos GIFs a SWF, soporta GIFs animados.
- **WAV2SWF** Convierte archivos de audio en formato WAV a SWFs, usando la librería L.A.M.E. MP3.
- **AVI2SWF** Convierte archivos en formato AVI a SWF. Soporta compresión Flash MX H.263.
- **Font2SWF:** Convierte archivos de Fuente (TTF, Type1) a SWF.
- **SWFC:** Permite generar archivos SWF a partir de scripts.
- **SWFExtract:** Permite extraer MovieClips, Sonidos, Imágenes, y otros elementos, de una película Flash.
- **RFXSWF.** Es una librería que puede ser usada para la generación estándar de archivos SWF. Incluye soporte para Mapas de Bits, Botones, Formas, Texto, Sonido, etc. También admite ActionScript usando las librerías Ming para generar SWF.

(Rainer Böhme & Matthias Kramm 2007)

### 1.3 Descripción General del Objeto de Estudio

Actualmente el desarrollo de productos multimedia en la Universidad se realiza utilizando software propietario, principalmente Adobe Flash 8. Este software brinda diversas facilidades para el desarrollo rápido de este tipo de aplicaciones, entre otras cosas ofrece un ambiente amigable para el diseño y animación de la interfaz, así como un editor de código ActionScript, sin embargo los problemas que enfrenta Cuba por el injusto bloqueo que mantiene Estados Unidos desde hace más de 40 años, impide la adquisición de las licencias necesarias para producir y comercializar estos productos, y en caso de poder adquirirlas se haría económicamente imposible debido al alto costo de las mismas.

Por otra parte, la estrategia de migración a software libre que se está llevando a cabo a nivel nacional hace imposible que se continúe utilizando esta herramienta, para lo cual se realiza la propuesta de un IDE para el desarrollo de software educativo multimedia, y como parte fundamental y objetivo del presente documento: Desarrollo de un Editor de Código que permita y facilite la gestión de código en el ambiente de Desarrollo Integrado (IDE) propuesto.

### 1.4 Identificación de la audiencia

El software desarrollado irá dirigido a todas aquellas personas que desarrollan productos multimedia en la Universidad, aunque es extensible a todo el país y también a nivel internacional, ya que brindará un conjunto de facilidades que permitirán escribir código específico para multimedia de forma eficiente y rápida. Para el uso de la misma es necesario conocer algún lenguaje de programación para multimedia, por ejemplo ActionScript, y tener conocimientos de programación orientada a objetos.

### 1.5 Conceptos Generales relacionados

- Multimedia: Aplicación que integra varios recursos comunicativos y audiovisuales para comunicar determinado conocimiento, y permite al usuario interactuar con el contenido del mismo.
- ActionScript: Lenguaje de scripts muy parecido al JavaScript, pero diseñado específicamente para la creación de multimedia interactivas utilizando Flash. Comenzó por la versión 1.0 y debido al enorme éxito se encuentra en la versión 3.0, es Orientado a Objetos semejante a los lenguajes de programación tradicionales como C++ y Java.
- IDE: Ambiente de Desarrollo Integrado (Integrated Development Enviroment). Un IDE es un programa que integra un conjunto de herramientas para el desarrollo de software, automatizando las tareas que repite a menudo como compilar, construir, ejecutar, creación de nuevos proyectos,

configuración de opciones, entre otras. Además brindan facilidades al proceso de edición de código resaltando las palabras claves del lenguaje y permitir el autocompletamiento.

- Editor de Código: Es un programa que permite editar código escrito para determinado lenguaje, generalmente ejecuta el coloreado de sintaxis de las palabras claves del lenguaje, sin embargo no cuenta con herramientas para la compilación y construcción de proyectos.
- Licencias de software: especie de contrato donde se especifican las normas y reglas para utilizar determinado software, entre estas normas están los permisos de copia, distribución y modificación del programa. Existen muchos tipos de licencias, sin embargo se dividen generalmente en dos grupos: privativas y libres.
- Software Libre: se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades fundamentales de los usuarios del software planteadas por la Fundación de Software Libre (FSF), las cuales son: libertad de usar el programa para cualquier propósito, libertad de estudiar cómo funciona el programa y adaptarlo a sus necesidades, la libertad de distribuirlo, la libertad de mejorar el programa y hacer públicas las mejoras a los demás, logrando un beneficio a la comunidad (Free Software Foundation 2008).
- Software Privativo: se refiere a todo aquel software que no cumple con alguna de las 4 libertades fundamentales planteadas por la FSF, entre ellos se encuentran los Contratos de Licencia de Acuerdo Final del Usuario, que limita enormemente los derechos de un usuario.

## 1.6 Tendencias y Tecnologías

El mundo de la informática se desarrolla vertiginosamente. El software como parte de él evoluciona, y surgen así nuevas metodologías y formas de desarrollo, que posibilitan la creación de productos cada vez más complejos.

Con la aparición de los primeros programas con interfaz gráfica de usuario (GUI) la informática dio un paso gigantesco en su desarrollo, acercándose mucho más al usuario y mejorando la experiencia del usuario utilizando determinado software. Actualmente para el desarrollo de interfaces visuales existen muchas librerías gráficas, algunas son multiplataformas como GTK y QT, otras son específicas de determinado sistema. Una librería gráfica provee un conjunto de clases y componentes que permiten el desarrollo rápido de interfaces visuales, en este capítulo se abordan las más importantes en el siguiente epígrafe.

Como parte de este desarrollo, el mundo de la programación vivió el surgimiento de nuevos lenguajes cada vez más potentes, y nuevas metodologías de desarrollo, así como nuevos paradigmas de programación, causando la Programación Orientada a Objetos uno de los mayores impactos, al permitir abstraer problemas del mundo actual a la programación.

Para desarrollar esta propuesta, es necesario que el software cuente con una interfaz gráfica de usuario que permita de manera sencilla y rápida realizar diferentes operaciones, para lo cual se debe seleccionar una librería gráfica específica que permita elaborar un sistema robusto, estable, y multiplataforma. También se necesita una metodología que guíe el proceso de desarrollo del software y un lenguaje de programación, que permitirá la implementación del producto.

## 1.7 Librerías Gráficas.

### 1.7.1 GTK

Gtk es un conjunto de librerías gráficas escritas en C que proveen un conjunto de componentes que permiten la construcción de modernas interfaces visuales para aplicaciones de escritorio. Gtk fue desarrollada inicialmente como un conjunto de herramientas libres para GIMP, un programa de edición de imágenes, pero posteriormente fueron extendidas al escritorio.
GTK se distribuye bajo licencia LGPL, y cuenta con suficiente documentación electrónica, aunque mayormente en inglés e incluye un constructor de interfaces visuales llamado Glade, y comprende las librerías: Glib, GdkPixbuf, Gdk, Gtk, Atk y Pango. (The Gtk Team 2008)

Gtkmm no es más que la misma versión de Gtk pero portada a C++. Debido a las ventajas de la Programación Orientada a Objetos (POO) y a que C++ a diferencia de C está orientado a objetos destacaremos las funcionalidades y ventajas de Gtkmm como librería.

Como características fundamentales de esta librería podemos mencionar:

- Es multiplataforma.
- El lenguaje de programación utilizado es C++, con soporte para el sistema de señales y slots (SIGNALS / SLOTS).
- La estructura interna de la librería es la de una jerarquía de clases formada por varios árboles que poseen distintas raíces a través de herencia simple.

Son productos estables, los cuales se encuentra en la versión 2.4.x de desarrollo.

### 1.7.2 QT

Qt es un producto de la empresa noruega de software Trolltech, esta empresa se dedica a desarrollar librerías y herramientas de desarrollo de software, además es experta en servicios de consultoría. Qt es un conjunto de librerías para el desarrollo de aplicaciones GUI, escritas en código C++, y son completamente orientadas a objetos. Estas librerías comenzaron a distribuirse comercialmente en 1996 y desde entonces han sido la base para numerosas aplicaciones, incluyendo la popular interfaz gráfica para Linux llamada KDE, disponible en todas las grandes distribuciones de Linux (Jasmin Planchette & Mark Summerfield 2006). Actualmente ha evolucionado hasta convertirse en un completo framework para el desarrollo de aplicaciones.

En el mercado se pueden encontrarlas siguientes distribuciones de Qt:

• *Qt Enterprise Edition* y *Qt Professional 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 and the 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* es una versión para dispositivos móviles como celulares.

Características:

- QT es una librería para la creación de interfaces gráficos. Se distribuye bajo una licencia libre GPL(o QPL) que permite incorporar QT4 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.
- Es orientado a objetos, lo que facilita el desarrollo de software. El lenguaje para el que se encuentra disponible es C++ aunque han aparecido bindings a otros lenguajes como Python.
- Es una librería que se basa en los conceptos de widgets (objetos), Señales-Slots y Eventos (ej: clic del ratón).
- Las señales y los slots es el mecanismo para que unos widgets se comuniquen con otros.
- Algunos atributos como el texto de etiquetas y otros se modifican de modo similar al lenguaje HTML
- QT proporciona además otras funcionalidades:
  - Librerías básicas para Entrada/Salida, Manejo de Red, XML
  - Interfaces con bases de datos como Oracle, MySQL, PostgreSQL, ODBC.
  - Plugins, librerías dinámicas (Imágenes, formatos,...)

### 1.7.3 wxWidgets

wxWidgets son un conjunto de librerías gráficas creadas en 1992 como un proyecto para realizar aplicaciones multiplataformas para entornos Unix y Microsoft Windows, luego como parte de su evolución fue añadido soporte para otras plataformas como WinCE y Mac .
wxWidgets provee una forma sencilla de escribir aplicaciones con interfaz visual para múltiples plataformas utilizando los controles nativos del sistema donde se esté ejecutando. Cuenta con todas las funcionalidades de las más modernas librerías gráficas que existen: programación para redes, multihilo, manejo de imágenes, soporte para base de datos, entre otras (Julian Smart, Kevin Hock & Stefan CSomor 2006).

Características:

- *Multiplataforma:* permite crear código multiplataforma para la mayoría de los sistemas operativos existentes (Microsoft Windows 95/98/ME/NT/2K/XP, GNU/Linux y Unix, MacOS).
- *Open Source:* wxWidget es software libre y abierto, desarrollado por un enorme grupo de desarrolladores entusiastas y talentosos a través de Internet y cualquiera puede acceder y modificar su código.
- *Documentación y Ejemplos:* posee un manual de referencia con cerca de 800 páginas en formato HTML, Windows Help, y Acrobat PDF.
- *Múltiples controles y clases de interfaz:* wxWidgets posee los controles básicos usuales en toda librería gráfica como botones, cajas de texto, checkbox, combobox, etcétera, además de un conjunto de clases avanzadas como wxTreeCtrl, wxNotebook, wxMDIParentFrame, que permiten la construcción de modernas interfaces visuales.
- *Poderoso sistema de eventos:* posee un sistema de eventos similar a los mapas de mensajes MFC, que permite asociar eventos con funciones miembros ya sea estáticamente (en tiempo de compilación) o dinámicamente (en tiempo de ejecución). A diferencia de QT no requiere un procesador no estándar. También pueden definirse nuevos eventos.
- *Facilidades para Impresión:* provee facilidades de impresión, utilizando Windows Printing en Microsoft Windows, y generando PostScript en Unix.
- *Compiladores:* wxWidgets soporta mucho más compiladores que posiblemente cualquier otro framework, incluye la mayoría de todos los compiladores populares de C++ en Microsoft Windows, excepto Symantec C++, y todos los de Unix, incluyendo MINGW32 y G++.
- *Base de Datos:* contiene un set de clases ODBC que permiten conectar y manejar diferentes motores de base de datos como MySQL, Postgress, y la popukar SQLite.

- *Multihilo:* igual que muchos frameworks, posee un conjunto de clases que facilitan la programación de aplicaciones multihilo, entre ellas clases semáforos y un grupo de hilos que deben ser utilizadas cuidadosamente por el programador. Estas clases aún están en desarrollo.
- *Red:* posee un completo set de clases TCP/IP, incluyendo soporte a los protocoles FTP y HTTP.
- *Clases HTML:* este conjunto de clases permiten visualizar e imprimir documentos HTML.
- *Integración con OpenGL:* el paquete wxGLCanvas permite soportar OpenGL realizando pequeños cambios en aplicaciones OpenGL puras.
- *Soporte de Diagramas en aplicaciones:* si es necesario implementar una aplicación CASE ( Rational Rose, Visual Paradigm ) , o el modelado de una red, o alguna que requiera la utilización de nodos y arcos, es posible utilizar la librería Object Graphics Library (versión 3.0), facilitando el trabajo.

## 1.8 Lenguajes de Programación

### 1.8.1 Java

Java se diseñó para abordar las dificultades que plantea el mundo moderno de la computación distribuida basada en la web: la creación de aplicaciones portátiles, seguras, conectadas en red. Java no es sólo un C++ mejorado con una biblioteca de redes; también es un ambiente completo para producir aplicaciones distribuidas, a efecto de que los programas de Java se ejecuten en forma segura e idéntica en diversos tipos de computadoras (Jeffrey Savit, Sean Wilcox & Bhuvana Jayaraman 1999).

Características fundamentales:

- Multiplataforma: Java es multiplataforma, es decir, el código generado por un compilador Java, conocido como “ByteCodes”, puede ser ejecutado en gran variedad de tipos de computadores sin recompilar ni cambiar un solo byte. Los bytecodes deben ser interpretados por una máquina virtual (JVM, Java Virtual Machine). Esta máquina virtual es una aplicación que debe existir en toda plataforma en donde se desee ejecutar una aplicación Java. Actualmente la mayoría de todos los sistemas operativos poseen una implementación oficial de la JVM.

- Lenguaje poderoso y moderno: La sintaxis del Java está basada en C++. Aunque no se puedan manejar los punteros directamente, se puede hacer uso de referencias que brindan casi la misma funcionalidad. Además posee un colector de basura que permite gestionar la memoria más fácilmente.
- Aplicaciones seguras: Las aplicaciones web Java (applets) son seguras, en cuanto a que no pueden acceder a recursos internos de la PC (Ej: disco rígido) sin que se le brinden privilegios específicamente. Además un applet descargado desde un servidor no puede conectarse a otro.

Desventajas:

Aplicaciones poco eficientes: el hecho que las aplicaciones Java deban ser interpretadas por otra aplicación situada entre el programa y el sistema operativo, resulta en una disminución de rendimiento considerable. Además el uso de la memoria en aplicaciones Java es por lo general mayor.

JVM: es la máquina virtual de Java, el software que interpreta el bytecode debe estar debidamente instalado en todo aquel sistema donde se vaya a correr una aplicación Java. A pesar de estos inconvenientes, cuando se necesita producir un software multiplataforma, Java constituye una fuerte opción para lograrlo.

### 1.8.2 C++

C + + fue inventado en 1980 por Bjarne Stroustrup en los Laboratorios Bell en Murray Hill, New Jersey. Inicialmente recibió la denominación de C con clase. En 1983 se le dio el nombre de C + +. Desde 1980 C++ ha sufrido dos importantes revisiones, una en 1985 y otra en 1990. En la actualidad se está trabajando para elaborar un estándar ANSI (American National Standards Institute) para C+ + (Herbert Schildt 1995).

Entre las principales características añadidas al C++ que lo hacen una extensión del C se encuentran: la POO y la programación genérica, permitiendo el uso de plantillas.

El C++ ha sido utilizado ampliamente en aplicaciones que necesitan un rendimiento optimizado debido a su magnitud, así como en el desarrollo de la mayoría de los sistemas operativos actuales.

Características fundamentales:

- *Programación orientada a objetos:* La posibilidad de orientar la programación a objetos permite al programador diseñar aplicaciones desde un punto de vista más cercano a la vida real, y permite la reutilización del código de una manera más lógica y productiva.
- *Portabilidad:* Un código escrito en C++ puede ser compilado en casi todo tipo de ordenadores y sistemas operativos sin hacer apenas cambios.
- *Brevedad*: El código escrito en C++ es muy corto en comparación con otros lenguajes, sobre todo porque en este lenguaje es preferible el uso de caracteres especiales que las "palabras clave".
- *Programación modular*: Un cuerpo de aplicación en C++ puede estar hecho con varios ficheros de código fuente que son compilados por separado y después unidos. Además, esta característica permite unir código en C++ con código producido en otros lenguajes de programación como Ensamblador o el propio C
- *Velocidad*: El código resultante de una compilación en C++ es muy eficiente, gracias a su capacidad de actuar como lenguaje de alto y bajo nivel y a la reducida medida del lenguaje.

- *Potencia***:** Permite trabajar tanto a bajo como a alto nivel, permitiendo un mayor control sobre el manejo de la memoria.
- *Punteros***:** la utilización de punteros permite asignar dinámicamente y reservar espacios de memoria, mejorando la velocidad de la aplicación.
- *Sobrecarga de Operadores*: permite redefinir operadores como los unarios (suma, resta, etcétera) para realizar operaciones con tipos de datos más complejos.
- *RTTI:* es capaz de identificar tipos de datos en tiempo de ejecución.

Desventajas:

La posibilidad de trabajar tanto a alto como a bajo nivel así como la ausencia de automatismos que proveen otros lenguajes más modernos como Java o C#, obliga a hacer casi todo manualmente igual que en C, trae como consecuencia que sea difícil de aprender.

### 1.8.3 C#

C# es un lenguaje orientado a objetos desarrollado por la empresa Microsoft como parte de su plataforma .NET, el cual fue aprobado como un estándar por la ECMA e ISO. Su sintaxis deriva de C++ y utiliza el modelo de objetos de la plataforma .NET que similar a la de JAVA, aunque incluye mejoras también tomadas de otros lenguajes como Delphi.

C# amplia las capacidades de C, C++, Visual Basic (VB) y Java para proporcionar un completo entorno de desarrollo en el que crear aplicaciones. C# mezcla la potencia de C, las capacidades de orientación a objetos de C++ y la interfaz gráfica de Visual Basic. (Geetanjali Arora, Balasubramaniam Aiaswamy & Nitin Pandey 2002)

Características:

- *Sencillez:* C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios.
- *Modernidad:* C# incorpora en el propio lenguaje elementos que a lo largo de los años ha ido demostrándose son muy útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción foreach que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico string para representar cadenas o la distinción de un tipo bool específico para representar valores lógicos.
- *Orientación a objetos:* Como todo lenguaje de programación de propósito general actual, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del CTS que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el

de C# es más puro en tanto que no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código. C# soporta todas las características propias del paradigma de programación orientada a objetos: encapsulación, herencia y polimorfismo.

- *Orientación a componentes:* La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros)
- *Gestión automática de memoria:* Como ya se comentó, todo lenguaje de .NET tiene a su disposición el recolector de basura del CLR. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos. Sin embargo, dado que la destrucción de los objetos a través del recolector de basura es indeterminista y sólo se realiza cuando éste se active, ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente, C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción using.
- *Seguridad de tipos:* C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evita que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura.
- *Sistema de tipos unificado:* A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada System.Object, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán "objetos")
- *Compatible:* Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados Platform Invocation Services (PInvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32. Nótese que la capacidad de usar punteros en código inseguro permite que se pueda acceder con facilidad a este tipo de funciones, ya que éstas muchas veces esperan recibir o devuelven punteros.

Desventajas:

A pesar de la facilidad que brinda C# para el desarrollo de aplicaciones, el principal inconveniente es que las aplicaciones desarrolladas en este lenguaje se ejecutan sobre una máquina virtual, en este

caso el Framework .NET de Microsoft, impidiéndole competir en velocidad y rendimiento con lenguajes compilados como C/C++.

Otro punto en desventaja es que el Framework .NET solo se encuentra disponible para la plataforma Microsoft Windows, por lo que las aplicaciones desarrolladas solo estarán disponibles para ese sistema. Actualmente existe el proyecto Mono, el cual lleva a cabo una implementación libre del framework .NET, así como de todas las librerías de C#, pero aún no se encuentra completo.

### 1.9 UML. El Lenguaje de Modelado Unificado

Inicialmente, cuando el mundo de la computación se encontraba aún en los inicios, se escribían programas que por su baja complejidad no requerían un análisis tan profundo, de este modo los programadores realizaban un análisis mínimo antes de comenzar a escribir código, y los cambios necesarios eran hechos directamente sobre la marcha del proceso de desarrollo del software o al final, en forma de parches o actualizaciones. También el número de programadores desarrollando el mismo producto era bastante bajo, generalmente dos o tres, por lo que la coordinación entre ellos estaba bastante organizada.

Con el transcurso de los años, unido al auge de la informática, los sistemas informáticos fueron tornándose cada vez más complejos, así como las necesidades de las empresas que requerían determinados programas aumentaron en requerimientos y complejidad. La tarea de desarrollar software se tornó complicada, era necesario encontrar un lenguaje que permitiera estructurar y representar visualmente un análisis del problema, un acercamiento a las clases que se implementarían así como colaboraciones entre ellas, entre otros aspectos. Los equipos de desarrollo también habían aumentado su número, haciéndose necesario cada vez más un lenguaje común entre todos los miembros del equipo para comprender el problema y llegar a una solución conjunta del problema. Surge así el Lenguaje Unificado de Modelado (UML Unified Model Language).

El UML es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson, los cuales trabajaban en empresas diferentes durante la década de los años 80 y principios de los 90 y cada uno había diseñado su propia metodología para el análisis y diseño orientado a objetos. A mediado de los 90 comenzaron a intercambiar ideas y decidieron desarrollar su trabajo en conjunto. Los anteproyectos de UML empezaron a circular en la industria del software y las reacciones resultantes trajeron consigo considerables modificaciones. A medida que muchos corporativos vieron que UML era útil a sus propósitos se formó un consorcio del UML, entre ellos destacan DEC, Hewlett-Packard, Intellicorp,

Microsoft, Oracle, entre otros. En 1997 el consorcio produjo la versión 1.0 del UML y lo presentó al Grupo de Administración de Objetos (OMG) como propuesta para un lenguaje de modelado estándar. A continuación en los años siguientes aparecieron nuevas versiones, convirtiéndose el lenguaje UML en un estándar de facto en la industria del software, el cual continúa evolucionando (Joseph Schmuller 1999).

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. Los principales y más conocidos son:

- Diagrama de clases.
- Diagrama de objetos
- Diagrama de Casos de Uso

A través de UML podemos desarrollar un diseño sólido y a la vez flexible de la aplicación.

## 1.10 RUP como metodología para el desarrollo.

Las metodologías de desarrollo son un conjunto de procedimientos, técnicas y normas que ayudan el proceso de desarrollo de software así como su documentación. Existen diferentes metodologías para hacer un software, y cada una se adapta a determinada situación, sin embargo, para proyectos grandes y complejos muchos autores recomiendan el uso del Proceso Unificado de Desarrollo de Software (RUP).

RUP (Rational Unified Process) divide en 4 fases el desarrollo del software:

- Inicio, El Objetivo en esta etapa es determinar la visión del proyecto.
- Elaboración, En esta etapa el objetivo es determinar la arquitectura óptima.
- Construcción, En esta etapa el objetivo es llevar a obtener la capacidad operacional inicial.
- Transición, El objetivo es obtener una versión funcional 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. Vale mencionar que el ciclo de vida que se desarrolla por cada iteración, es llevada bajo dos disciplinas:

- Disciplina de Desarrollo:
  - Ingeniería de Negocios: Comprender las necesidades del negocio.

  - Requerimientos: Trasladar las necesidades del negocio a un sistema automatizado
  - Análisis y Diseño: Trasladar los requerimientos dentro de la arquitectura de software.
  - Implementación: Crear un software que se ajuste a la arquitectura y que tenga el comportamiento deseado.
  - Pruebas: Asegurar que el comportamiento requerido es el correcto y que todo los solicitado está presente.
- Disciplina de Soporte
  - Configuración y administración de cambios: Guardar todas las versiones del proyecto.
  - Administrando el proyecto: Administrar horarios y recursos.
  - Ambiente: Administrar el ambiente de desarrollo.
  - Distribución: Hacer todo lo necesario para la salida del proyecto

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierte luego en un entregable al cliente. Esto trae como beneficio 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.

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.

## 1.11 IDEs para el Desarrollo

### 1.11.1 KDevelop

Es un Entorno de Desarrollo Integrado (IDE) para sistemas Unix, publicado bajo licencia GPL. Su desarrollo comenzó en el año 1998 con el fin de desarrollar un IDE fácil de usar para el entorno de escritorio KDE, anunciándose por primera vez en diciembre del 2003, junto al lanzamiento de KDE 3.2 (Miguel Ángel De Blas Búrdalo 2008). A partir de la versión 3.0 fue reconstruido completamente para la corrección de problemas e incorporación de muchas mejoras. KDevelop a diferencia de otras IDEs de desarrollo no posee un compilador propio, por lo que depende de gcc para generar código binario.

Características:

- Editor de código fuente con resaltado de sintaxis e indentado automático.
- Soporte para múltiples lenguajes ( C++, Ruby, Java, ADA, Pascal )
- Gestión para diferentes tipos de proyectos, como Automake, Qmake (para QT), y Ant (para proyectos JAVA).
- Frontend para gcc.
- Frontend para el depurador de GNU.
- Asistentes para generar y actualizar las definiciones de clases y el framework de la aplicación.
- Asistentes para la creación de diferentes tipos de proyecto.
- Completamiento de código C y C++.
- Compatibilidad nativa con el generador de documentación Doxygen.
- Soporta diferentes sistemas de control de versiones ( SVN, CVS )
- Integración con QT, permitiendo utilizar el diseñador de interfaces QT Designer para diseñar la GUI de la aplicación.

### 1.11.2 CodeBlocks

Codeblocks es un completo IDE multiplataforma para C++ desarrollado utilizando las librerías gráficas wxWidgets. Es altamente extensible a través de plugins y contiene un gran conjunto de opciones que permiten personalizarlo según las necesidades. Está disponible bajo licencia GPL. (The Code::Blocks Team 2008)

Características:

- Soporte para múltiples compiladores. ( GCC, MSVC++, Digital Mars, Borland C++ 5.5, OpenWatcom )
- Sistema de construcción de proyectos personalizado y rápido (no necesita MAKEFILES).
- Proyectos para múltiples plataformas.
- Permite importar proyectos de Microsoft Visual C++
- Permite depurar el código utilizando GNU GDB.
- Resaltado de sintaxis.
- Indentación de código.
- Edición de múltiples archivos a través de pestañas (tabs).
- Completamiento de código.
- Explorador de clases.
- Personalización de herramientas externas.

### 1.11.3 Eclipse

Eclipse es un poderoso IDE desarrollado inicialmente por IBM para el desarrollo de aplicaciones utilizando Java, actualmente su desarrollo es llevado a cabo por la Fundación Eclipse, una organización independiente y sin ánimos de lucro que fomenta una comunidad de código abierto, así como una serie de servicios.

La base para Eclipse es la Plataforma de Cliente Enriquecido, en inglés conocida como RCP (Rich Client Plataform). Una Plataforma de Cliente enriquecido, a diferencia de las aplicaciones Cliente-liviano basadas en navegadores, se construye a partir de un conjunto de componentes que brindan determinadas funcionalidades (The Eclipse Foundation 2008).

Eclipse está formado por:

- Plataforma principal, inicio de Eclipse, ejecución de plugins.
- Standard Widget Toolkit (SWT), un conjunto de componentes para desarrollar interfaces visuales, desarrollados por la propia fundación.
- Jface, para el manejode archivos, texto, editores.
- Workbench de Eclipse, un conjunto de vistas, perspectivas, asistentes.

Eclipse se encuentra basado en plugins para brindar todas sus funcionalidades, a diferencia de otros entornos monolíticos que incluyen todas las funcionalidades, las necesite el usuario o no, y permite extenderse a otros lenguajes como C++ y PHP, agregándole el plugin correspondiente.

Las versiones actuales de Eclipse cuentan con las siguientes características:

- Editor de texto
- Resaltado de sintaxis
- Compilación en tiempo real
- Pruebas unitarias con JUnit
- Control de versiones con CVS
- Integración con Ant
- Asistentes (wizards): para creación de proyectos, clases, tests, etcétera.
- Refactorización

### 1.11.4 Anjuta

Anjuta es un versátil IDE para C/C++ desarrollado para GNU/Linux, específicamente para la creación de aplicaciones GTK y GNOME, proporcionando un conjunto de funcionalidades que agilizan el proceso de implementación de un software, entre ellas: Administración de archivos del proyecto, Asistentes para la creación de nuevos proyectos, debugger, editor de código con autocompletamiento y resaltado de sintaxis.

Actualmente, aunque se continúa el mantenimiento de la última versión estable (1.2), la versión 2 tiene importantes mejoras entre las que destaca:

- nuevo sistema de extensiones, todos los de la primera versión son compatibles.
- arquitectura revisada y extensible.
- nuevo Intérprete de comandos propio y documentación del API.
- integrado un nuevo sistema de ayuda.
- un diseñador gráfico de interfaces de usuario (todavía incompleto) con Glade.
- mejoras diversas en el editor de programación (edición remota, mejor realce de sintaxis, etcétera).
- nuevo administrador de tareas.
- extensión para añadir macros, insertar texto predefinido o personalizado.
- plantilla fácilmente extensible para proyectos mediante asistente.
- extensión para Subversion (todavía incompleto)
- administrador de sesiones de trabajo.
- actualizado la extensión para CVS.
- y otras diversas mejoras o actualizaciones.

## 1.15 Librería seleccionada para la Interfaz Visual.

Teniendo en cuenta todo el análisis referente al estado del arte de las librerías para la construcción de interfaces de usuario modernas, se decidió utilizar Qt4 para el desarrollo de la parte visual del software.

Los principales aspectos considerados fueron:

- Qt4 cuenta con versiones para casi todos los sistemas operativos existentes, como Linux, Unix, Mac OS, Microsoft Windows, Windows CE, Solaris, BSD, entre otros, lo que permitirá que el producto sea multiplataforma sin tener que realizar grandes cambios en el código.

- Es completamente orientado a objetos.
- Consta con una excelente documentación y un conjunto de herramientas para el desarrollo que agilizan el proceso de desarrollo.
- El equipo posee conocimientos previos de trabajo con Qt4, por lo que la curva de aprendizaje es menor.
- Es un completo framework para desarrollar aplicaciones, cuenta con módulos para programación multihilo, optimización, trabajo con XML.

### 1.16 Lenguaje seleccionado para la implementación

A pesar de que el C# es un lenguaje moderno y fácil de aprender, al estar ligado a la plataforma .NET, que solo se encuentra disponible para Microsoft Windows, fue completamente descartado. Existe una implementación libre del framework de Microsoft llamada Mono, sin embargo aún se encuentra en desarrollo.

Java por su parte es un lenguaje ampliamente utilizado en el desarrollo de aplicaciones multiplataformas, sin embargo su máquina virtual (JVM Java Virtual Machine) hace un uso excesivo de los recursos del ordenador, por lo que en computadoras no tan modernas como lo son la mayoría de las existentes en Cuba, pudiera ocurrir que la aplicación resultante no se ejecute de forma óptima.

Se decide entonces utilizar C++ para llevar a cabo la implementación, el C++ es un lenguaje compilado, se acerca bastante a la programación a bajo nivel, permitiendo al desarrollador un control más avanzado sobre los recursos del sistema, y la creación de aplicaciones que consuman un mínimo de memoria. Unido a esto las librerías Qt4 están desarrolladas en este lenguaje, confirmando una vez más que es posible desarrollar aplicaciones en C++ que sean multiplataformas.

### 1.17 IDE seleccionado para el desarrollo

Entre los varios IDEs estudiados, se selecciona KDevelop, por su mejor integración con la librería seleccionada (QT). Anjuta está dirigido a GTK y a desarrollos sobre el escritorio Gnome, Codeblock por su parte se orienta a wxWidgets.

KDevelop se integra completamente con las herramientas qmake y uic que brinda Qt4. Qmake es utilizado para la gestión de archivos del proyecto, el cual crea un archivo con extensión .pro con un registro de todos los archivos que componen el proyecto ( archivos cabeceras .h , archivos .cpp,

interfaces visuales .ui , recursos, etcétera ), luego genera un MAKEFILE, que no es más que un conjunto de reglas para construir el proyecto, en este último archivo se definen librerías específicas para compilar, tipo de ejecutable, opciones extras del compilador, por otra parte la herramienta UIC se encarga de traducir a lenguaje C++ los archivos xml generados por el Qt Designer, permitiendo su uso en el proyecto. Todo este proceso es realizado por KDevelop de forma transparente al desarrollo.

Otras de las ventajas que ofrece este IDE son: autocompletamiento y sugerencias de código, resaltado de sintaxis, explorador de clases del proyecto, administrador de recursos, edición de múltiples archivos, asistente para creación de clases, integración con el generador de documentación Doxygen.

### 1.18 Metodología utilizada

Como metodología de desarrollo se utilizará RUP, debido al nivel de complejidad que presenta la aplicación a desarrollar, principalmente por la arquitectura basada en plugins que permitirá extender las funcionalidades del software sin modificar el código base, todo esto requiere un diseño sólido y un análisis profundo.

Según algunos autores, los aspectos más definitorios de RUP se definen en tres frases: dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental. También utiliza el lenguaje UML y está basado en arquitectura de componentes (Ivar Jacobson, Grady Booch & James Rumbaugh 2000).

### 1.19 Herramienta CASE para el modelado.

Para el modelado de la solución se utilizará Visual Paradigm, herramienta profesional que soporta el ciclo de vida completo de desarrollo de un software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

Entre sus principales características se encuentran:

- Soporte de UML versión 2.1
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio.
- Documento Modelado colaborativo con CVS y Subversion
- Interoperabilidad con modelos UML2
- Código a modelo,

- Código a diagrama
- Ingeniería inversa Java, C++,
- Generación de código
- Modelo a código,
- diagrama a código
- Editor de Detalles de Casos de Uso -
- Entorno todo-en-uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso
- Diagramas de flujo de datos
- Generador de informes para generación de documentación
- Distribución automática de diagramas -
- Reorganización de las figuras y conectores de los diagramas UML

## 1.20 Conclusiones

En este capítulo se realizó un estudio sobre el estado del arte de las herramientas existentes en el mundo del software libre para la creación de software educativo con tecnología multimedia. Como resultado se pudo observar que la mayoría de estas herramientas no fueron desarrolladas completamente, por lo que carecen de muchas funcionalidades importantes, y otras como el MTASC son compiladores modo consola, haciendo engorroso su uso por parte del usuario.

Como metodología de desarrollo se seleccionó RUP y como herramienta CASE Visual Paradigm 6.0, apoyados en UML. Para la implementación del software se escogió C++ debido a su eficiencia y trabajo a bajo nivel y como IDE para el desarrollo KDevelop. Por último para la construcción de la interfaz visual se analizaron las tres principales librerías gráficas existentes en el software libre: GTK, wxWidgets y Qt4, escogiéndose esta última por sus posibilidades, buena documentación y capacidad multiplataforma. Una vez definidos estos aspectos se hizo posible pasar al siguiente capítulo donde se presenta la solución propuesta.

# CAPÍTULO 2

## Presentación de la solución propuesta

### 2.1 Introducción

En el presente capítulo se realiza un Modelo de Dominio para un mejor análisis de la solución a desarrollar. Se hace un énfasis especial en la metodología RUP para desarrollar dicho modelo, identificando los conceptos y clases asociados al dominio del problema.
También se identifican los requerimientos funcionales y no funcionales de la aplicación, para obtener los Casos de Usos que guiarán el proceso de desarrollo del software.

### 2.2 Modelo de Dominio

#### 2.2.1 Descripción del Modelo de Dominio

Debido a la baja estructuración de los procesos del negocio y a que la solución se encuentra centrada en herramientas y tecnologías informáticas, se utilizará un Modelo de Dominio para identificar los principales conceptos y objetos asociados al dominio del problema.
Los objetos del dominio representan cosas, parte o eventos que ocurren en el entorno del sistema que se desea representar, es por ello que utilizaremos un Diagrama de Clases de UML para representar estos objetos.

#### 2.2.2 Descripción general del sistema

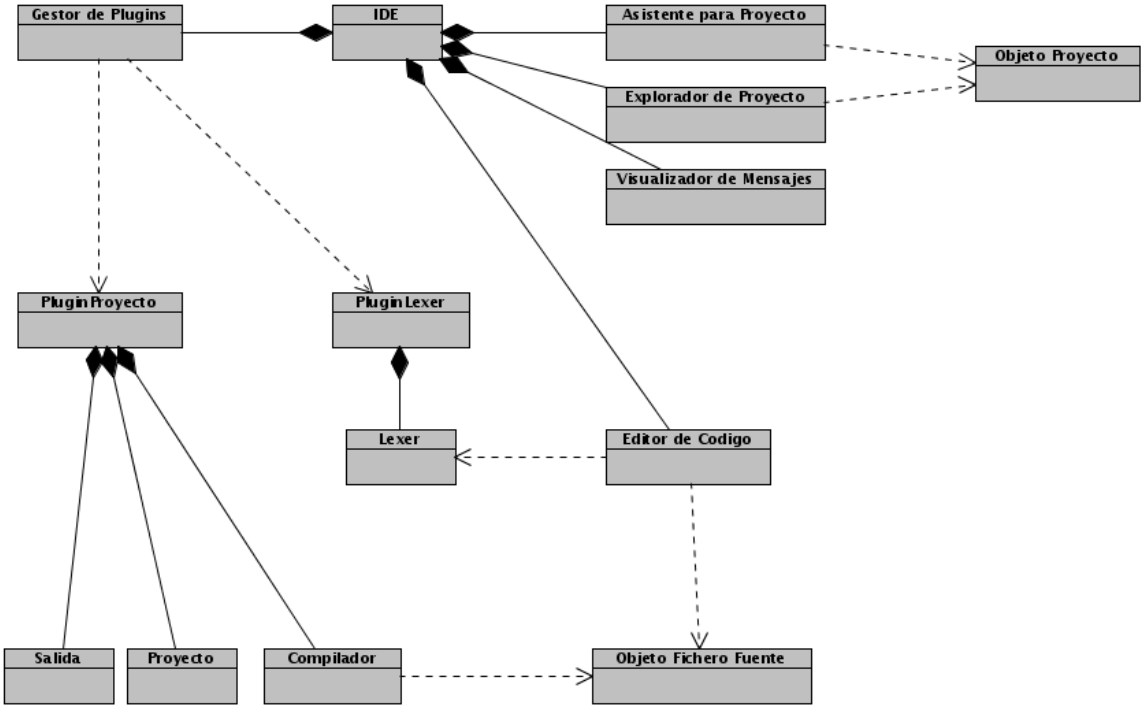
El sistema contará con una interfaz principal desde la cual el usuario tendrá acceso a todas las operaciones que permite el IDE ( Abrir Proyecto, Crear Proyecto, Guardar Archivo, entre otras), para ello contará con una barra de menú principal con los siguientes menús:

- Menú Proyecto: muestra las operaciones Abrir Proyecto, Crear Nuevo Proyecto, Cerrar Proyecto, Guardar Proyecto, Opciones de Proyecto.
- Menú Archivo: muestra las operaciones Abrir Archivo, Guardar Archivo, Cerrar Archivo.
- Menú Construir: muestra las operaciones Construir, Ejecutar, Detener.
- Menú Editar: muestra las operaciones básicas de cualquier editor de texto (Copiar, Cortar, Pegar, Seleccionar).

El sistema contará además con un sistema de plugins, que lo harán flexible a la hora de agregarle nuevas funcionalidades y nuevos compiladores. Este sistema trabaja de la siguiente forma:
Al iniciarse el programa el sistema busca en su carpeta de plugins los existentes y carga una referencia a ellos. Cuando el usuario selecciona en el menú Proyecto, la opción Nuevo Proyecto se lanza un asistente para la Creación de Nuevos Proyectos que lee la lista de proyectos disponibles, definidos en los plugins y los muestra, cuando el usuario selecciona uno entonces el sistema crea la estructura de archivos necesarios, y las opciones para dicho proyecto.

La interfaz principal estará construida usando paneles, en el panel superior se encontrarán los menú antes mencionados, a los laterales estará el panel Explorador de Clases y Archivos del Proyecto, en el centro y ocupando el mayor área de la interfaz estará el componente Editor de Código, donde el usuario efectúa todas las operaciones de edición de código, por último en la parte inferior se encuentra el panel Visualizador de Errores, donde se muestran los errores ocurridos durante la compilación/construcción del proyecto, así como otros mensajes del sistema.

## 2.3 Diagrama de Clases del Modelo del Dominio




## 2.4 Glosario de Términos del Dominio

A continuación un glosario de términos asociados al dominio, para una mejor comprensión del diagrama.

**IDE:** Es la aplicación principal, desde la cual el usuario puede realizar diversas acciones como abrir un proyecto, crear un nuevo proyecto, editar uno o varios archivos, compilar los ficheros fuentes, modificar las opciones del compilador del proyecto actual, etcétera. Gestiona de forma transparente las herramientas utilizadas para generar el ejecutable de la multimedia.

**Gestor de Plugins:** Es el encargado de detectar los plugins existentes en la carpeta de plugins del sistema y cargarlos.

**Asistente para Proyecto:** Permite crear proyectos de diferentes tipo de acuerdo al seleccionado por el usuario de una manera sencilla y rápida.

**Explorador de Proyecto:** Visualiza en todo momento todos los archivos pertenecientes al proyecto y las clases del proyecto.

**Visualizador de Mensajes:** Muestra todos los mensajes emitidos por el sistema cuando se encuentra algún proyecto abierto. También se incluyen los mensajes de error o advertencia emitidos por el compilador al construir la multimedia.

**Objeto Proyecto:** Es el archivo de configuración del proyecto, o sea, donde se guarda toda la información referente al proyecto, con una extensión determinada. La información del proyecto puede ser: Directorio de Salida, Ficheros en la biblioteca, entre otros. Es como decir el .FLA del Adobe Flash.

**Plugin:** Un plugin es un complemento, un fragmento de funcionalidad que se le agrega a un programa (si lo soporta) para añadirle funcionalidades, permitiendo nuevas opciones sin modificar el código base.

**Plugin Proyecto:** Agrega un nuevo proyecto a la lista de proyectos que el usuario puede crear a través del Asistente para Proyectos. Contiene información sobre cómo construir el proyecto, opciones, así como la estructura de directorios para ese proyecto. Permite agregarle nuevos proyectos al IDE sin modificar el código base.

**Plugin Lexer:** El plugin lexer permite que el IDE reconozca la sintaxis de nuevos lenguajes, incluso aunque no pueda compilarlos será posible editarlos con la comodidad de cualquier editor profesional.

**Salida:** Es el fichero resultante que se genera al compilar el sistema, el tipo de archivo y extensión depende del proyecto que se esté editando, y se define dentro del plugin de proyecto.

**Proyecto:** Es parte del Plugin de Proyecto, contiene información específica sobre determinado proyecto.

**Compilador:** Es parte del Plugin de Proyecto, permite la compilación de los ficheros fuente para generar el ejecutable o salida final.

**Lexer:** Un lexer define un conjunto de reglas para el resaltado de sintaxis de determinado lenguaje, es parte del Plugin Lexer.

**Editor de Código:** Permite editar uno o múltiples archivos, basado en un sistema de Tabs o Pestañas, a través del Plugin Lexer es capaz de reconocer la sintaxis del fichero en edición y realizar el resaltado de sintaxis, en el caso de tener un proyecto abierto, habilita el completamiento de código.

**Objeto Fichero Fuente:** es el fichero donde se guarda una clase, que luego es compilado a través del compilador. Ejemplos de ficheros fuente son .as (ActionScript), .h y .cpp (C++), .xml (XML), entre otros.

## 2.5 Levantamiento de Requisitos

### 2.5.1 Requisitos Funcionales

Los requisitos funcionales como su nombre lo indica son capacidades o funcionalidades que el sistema debe cumplir, se especifican por escrito, deben ser claros y precisos y posibles de probar o verificar.

**RF1.** El sistema debe soportar una arquitectura de Plugins que permita cargar dinámicamente la información de los diferentes tipos de ficheros que podrá soportar el sistema (ejemplo: *.as, *.hx, *.xml), así como el lexer de los lenguajes asociados a los ficheros.

**RF2.** El sistema debe soportar una arquitectura de Plugins que permita cargar dinámicamente los diferentes tipos de proyectos que soportará el sistema en dependencia del compilador que utilicen.

**RF3.** El sistema debe permitir *crear* un fichero de acuerdo a los tipos especificados en la arquitectura de Plugins.

**RF4.** El sistema debe permitir guardar un fichero que esté abierto y haya sido modificado (“Guardar”).

**RF5.** El sistema debe permitir guardar una copia del fichero con un nuevo nombre y ubicación (“Guardar como”).

**RF6.** El sistema debe permitir guardar todos los ficheros abiertos que hayan sido modificados de manera simultánea (“Guardar todo”).

**RF7.** El sistema debe permitir *abrir* un fichero de acuerdo a los tipos de ficheros especificados en los Plugins.

**RF8.** El sistema debe permitir cerrar un fichero seleccionado previamente por el usuario (“Cerrar”).

**RF9.** El sistema debe permitir cerrar de forma simultánea, todos los ficheros abiertos (“Cerrar todo”).

**RF10.** El sistema debe formatear el texto del fichero que esté editándose, según la información cargada mediante la arquitectura de Plugins (reconocimiento del léxico).

**RF11.** El sistema debe permitir crear un proyecto de acuerdo a los especificados en los Plugins.

**RF12.** El sistema debe permitir mostrar y modificar las opciones del proyecto que se encuentre abierto.

**RF13.** El sistema debe guardar en un fichero de configuración las opciones del proyecto abierto.

**RF14.** El sistema debe guardar en un fichero de configuración las opciones del compilador asociado al proyecto abierto.

**RF15.** El sistema debe guardar en un fichero de configuración las opciones de la salida final definida en el proyecto.

**RF16.** El sistema debe permitir abrir un proyecto.

**RF16.1** El sistema debe cargar de un fichero de configuración las opciones de proyecto,

**RF16.2** El sistema debe cargar de un fichero de configuración las opciones del compilador asociado al proyecto.

**R16.3** El sistema debe cargar de un fichero de configuración las opciones de la salida definida en el proyecto.

**RF17.** El sistema debe permitir seleccionar el tipo de salida deseada, en caso de que el proyecto soporte varias salidas.

**RF18.** El sistema debe mostrar y modificar las propiedades de la salida.

**RF19.** El sistema debe permitir mostrar y modificar las diferentes opciones del compilador.

**RF20.** El sistema debe permitir cerrar un proyecto.

**RF21.** El sistema debe brindar la opción de construir (Build Project) la salida del proyecto en edición.

**RF22.** El sistema debe brindar la opción de ejecutar la salida final del proyecto en edición (Run).

**RF23.** El sistema debe brindar la posibilidad de editar los comandos para la ejecución de la salida final del proyecto.

**RF24.** El sistema debe permitir agregar comandos para la construcción del proyecto.

**RF25.** El sistema debe visualizar los diferentes tipos de mensajes que se emiten por el compilador y por el propio sistema (mensajes de error, mensajes de alerta y cualquier otro tipo de mensajes).

**RF26.** El sistema debe visualizar los diferentes tipos de mensajes que emite la salida final del proyecto en tiempo de ejecución.

**RF27.** El sistema debe permitir salir completamente de la aplicación.

### 2.5.2 Requisitos No Funcionales

Los requerimientos no funcionales son cualidades o propiedades que tendrá el sistema, que definirán su comportamiento entre otras cosas. También se incluyen requerimientos de programas que deben encontrarse instalados para que el software funcione correctamente.

**Requerimientos de software**

Para la ejecución del software el sistema debe tener instalado Qt4.0 o superior.

**Requerimientos de interfaz**

La aplicación debe tener una interfaz sencilla e intuitiva, haciendo uso de barras de herramientas y menús que faciliten al usuario realizar las acciones comunes que se ejecutan en un Ambiente de Desarrollo Integrado. Debe contar con:

- Un asistente para la creación de proyectos.
- Un explorador de clases y archivos de proyecto, el cual se mostrará en forma de panel, en los laterales de la aplicación.

Además, debe garantizarse la mayor rapidez de respuesta del sistema frente a las acciones realizadas por el usuario sobre la interfaz visual.

**Requerimientos de Portabilidad**

La aplicación debe ser multiplataforma, pudiéndose ejecutar con cambios mínimos en el código sobre Microsoft Windows, GNU/Linux, y Mac OSX.

**Requerimientos Legales**

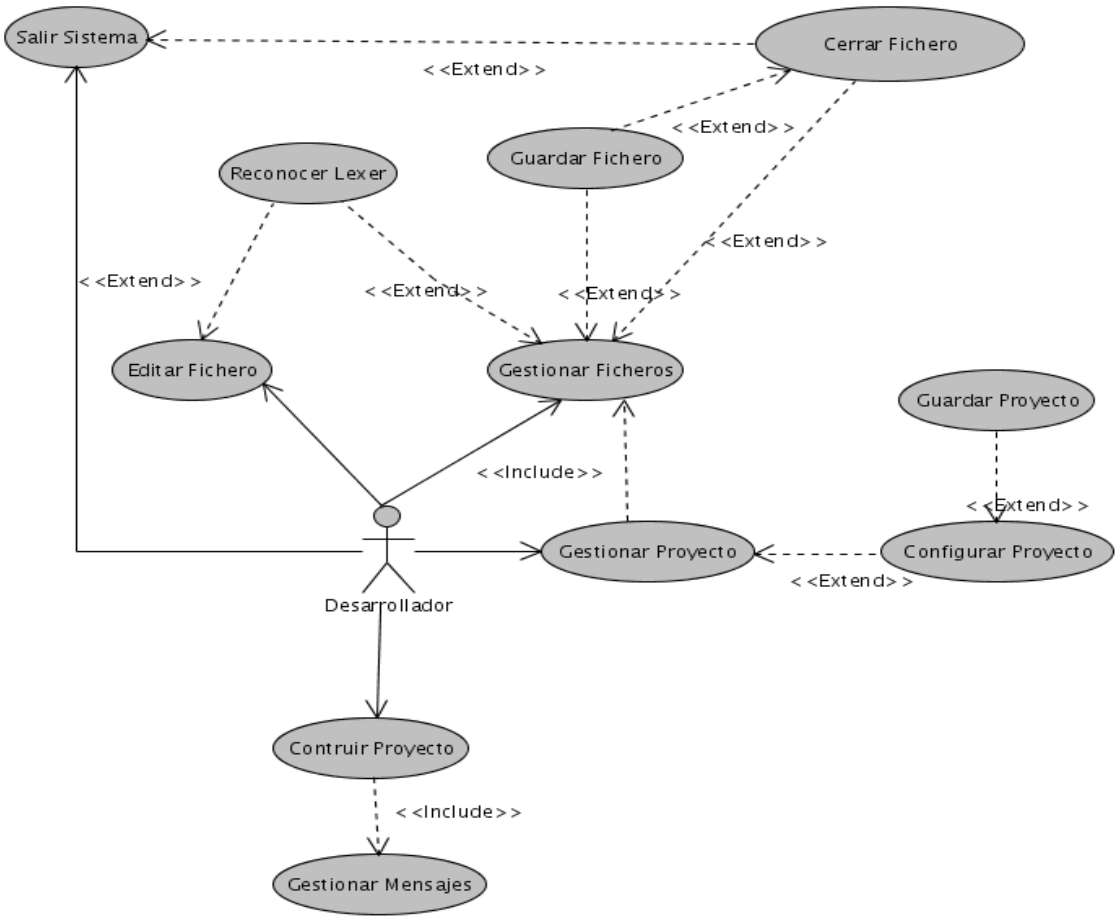
El sistema debe ser implementado utilizando herramientas libres, preferentemente bajo licencia GPL.

## 2.6 Actores y Casos de Uso del Sistema

### 2.6.1 Actores del Sistema

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

### 2.6.2 Diagrama de Casos de Uso del Sistema




## 2.7 Casos de Uso del Sistema

### 2.7.1 *Caso de Uso: Gestionar Ficheros*

El caso de uso se ejecuta cuando el desarrollador decide crear un nuevo fichero, abrir un fichero existente, cerrar un fichero, o cerrar todos los ficheros abiertos.

### 2.7.2 *Caso de Uso: Cerrar Fichero*

El caso de uso se ejecuta cuando el desarrollador decide cerrar un fichero.

### 2.7.3 *Caso de Uso: Guardar Fichero*

El Caso de Uso se ejecuta cuando el usuario selecciona alguna de las opciones para guardar el contenido del fichero que se encuentra editando. Estas opciones pueden ser: Guardar, Guardar como, y Guardar todo.

### 2.7.4 *Caso de Uso: Reconocer Lexer*

El Caso de uso es invocado por los casos de uso **Crear Fichero** o **Abrir Fichero,** y permite realizar el coloreado y resaltado de las palabras claves del lenguaje al que pertenece el archivo

### 2.7.5 *Caso de Uso: Salir del Sistema*

El Caso de Uso se inicia cuando el desarrollador selecciona la opción Salir, tanto en el Menú Archivo o presionando el botón Salir de la ventana principal.

### 2.7.6 *Caso de Uso: Gestionar Proyecto*

Este Caso de Uso se inicia cuando el desarrollador decide crear un nuevo proyecto, abrir un proyecto existente o cerrar el que esta editando.
Si el desarrollador presenta un proyecto abierto puede a su vez configurar las opciones del proyecto, del compilador y de la salida que se va a generar.

### 2.7.7 *Caso de Uso: Configurar Proyecto*

El caso de Uso de inicia cuando el desarrollador decide cambiar las opciones del proyecto.

### 2.7.8 *Caso de Uso: Guardar Proyecto*

El caso de Uso de inicia cuando el desarrollador una vez cambiado las configuraciones del Proyecto acepta dichos cambios.

### 2.7.9 *Caso de Uso: Construir Proyecto*

El caso de Uso de inicia cuando el desarrollador decide construir y ejecutar el Proyecto.
Construir el proyecto significa compilar todos los archivos fuentes del proyecto y generar a partir de estos un ejecutable que sería la aplicación final.

### 2.7.10 *Caso de Uso: Gestionar Mensajes*

El caso de uso es iniciado por el Caso de Uso Construir Proyecto de inicia cuando el desarrollador decide construir y ejecutar el Proyecto.

### 2.7.11 *Caso de Uso: Editar Fichero*

El Caso de Uso procede cuando el usuario introduce texto en el fichero abierto.

## 2.8 Conclusiones

El desarrollo del capítulo contribuyó a lograr una mejor comprensión del sistema que se desea construir, a través de una descripción de la solución propuesta y definiendo un listado de requerimientos funcionales y no funcionales que el sistema debe cumplir.

También se realizó el diagrama de dominio correspondiente, debido a la baja estructuración de los procesos del negocio, obteniéndose a través del Diagrama de Casos de Uso del Sistema las relaciones entre los actores y los 11 casos de uso presentes en el sistema, además se realizó la descripción de cada uno de ellos.

Con la finalización de este flujo de trabajo es posible transitar hacia la construcción de la solución propuesta.

# CAPÍTULO 3

## Construcción de la solución propuesta

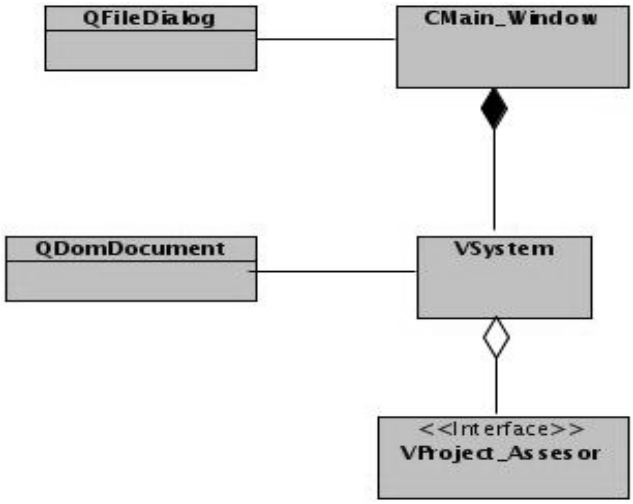
### 3.1 Introducción

En este capítulo se construye la solución propuesta a través de los flujos de trabajo de diseño e implementación. Se presenta el modelo de clases de diseño donde se exponen las realizaciones de los Casos de Uso descritos en el capítulo anterior. Además se presenta el modelo de despliegue y se explican los estándares utilizados para la implementación y para la construcción de la interfaz de usuario.

### 3.2 Diagramas de Clases del Diseño

El Diagrama de Clases de Diseño permite lograr una visión más detallada del conjunto de clases que se implementarán en el sistema, las relaciones existentes entre ellas y sus atributos. A continuación se hace un resumen de las principales clases del diseño de la aplicación. Para un mayor nivel de detalle consultar el Anexo B.

### 3.2.1 Clases del Diseño Abrir Proyecto




**CMain_Window:** Esta clase es la encargada de gestionar los eventos del usuario sobre la ventana principal de la aplicación así como los eventos emitidos por el menú principal y la barra de herramientas principal.

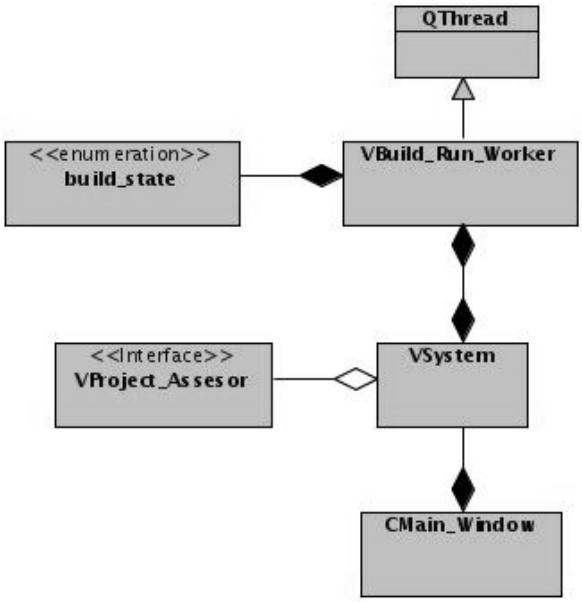
**VSystem:** Esta es la clase que se encarga de procesar todas las acciones importantes en la aplicación. Servira de intermediaria entre las interfaces y las clases del negocio, gestionando todos los eventos del sistema y delegando tareas a las clases correspondientes.

**VProject_Assessor:** Esta clase es la Interfaz del plugin de Proyecto, es la clase con la cual interactúa el sistema IDE directamente; es la cara del plugin al sistema IDE. Es una capa de abstracción que media entre el sistema IDE y las clases que contiene el plugin.

**QFileDialog:** Es una clase propia de las librerías QT. Provee un dialogo que le brinda la posibilidad al usuario de seleccionar archivos o directorios del disco duro del sistema.

**QDomDocument:** Esta es también una clase propia de las librerías QT. Representa básicamente un documento XML. Conceptualmente es la raíz del árbol del documento XML, y funcionalidades necesarias para acceder al todo el documento XML.

### 3.2.2 Clases del Diseño Construir Proyecto

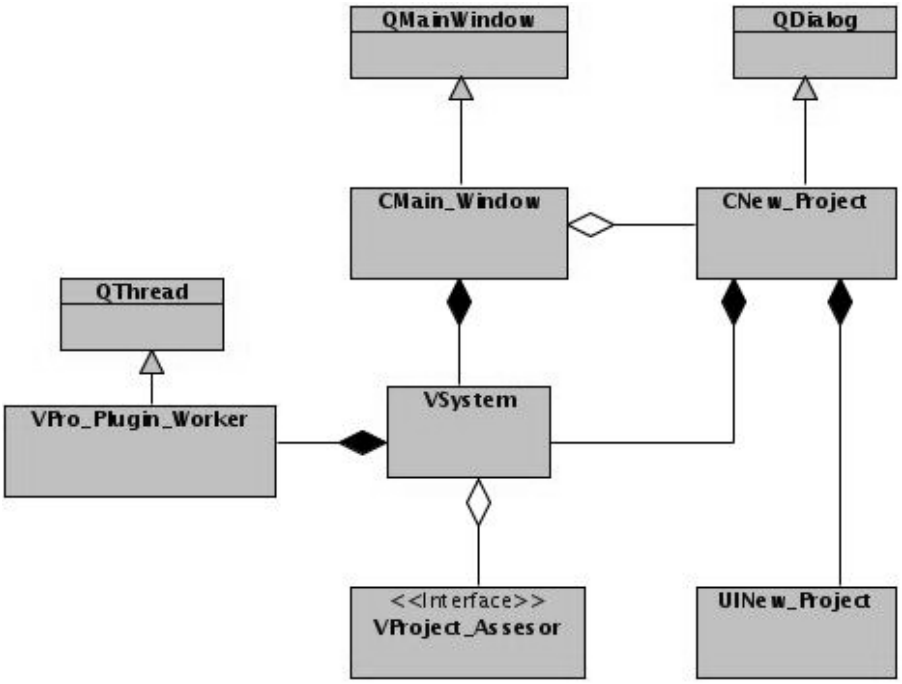



**VBuild_Run_Worker:** Esta clase es la encargada de realizar las tareas de Construir Proyecto, Ejecutar Proyecto. Por lo que presenta las funciones y atributos necesarios para tales fines. Hereda de QThread debido a que realiza la operación de Construcción del Proyecto en un hilo paralelo al hilo principal de la aplicación.

**QThread:** Es una clase propia de las librerías QT y permite la programación multihilo.

**build_state:** Es un ***enum*** que contiene los distintos estados en los que se encuentra el proceso de compilación-ejecución. Tales estados son: NORMAL (no se está realizando ninguna acción construir o ejecutar), RUNNING (la salida final esta ejecutándose), BUILDING (se está construyendo la salida final del proyecto). Este ***enum*** es utilizado por VBuild_Run_Worker para establecer, en cada momento de ejecución del sistema IDE, la acción que se encuentre realizando, permitiendo así que pueda ser consultado este estado siempre que sea necesario.

### 3.2.3 Clases del Diseño Nuevo Proyecto



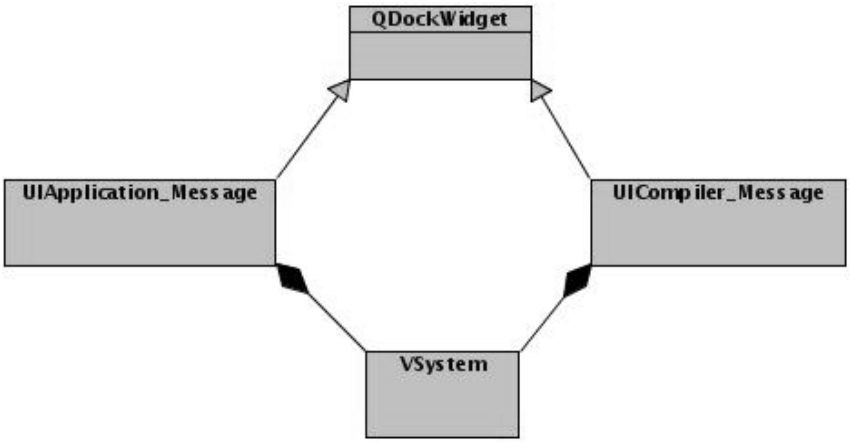

**QMainWindow:** Esta es una clase propia de las librerías QT. Provee la interfaz de una ventana principal de la aplicación. Con la barra de menú, barra de herramientas y barra de estado.

**UINew_Project:** Esta clase es la encargada de construir la interfaz grafica de usuario del asistente para nuevo proyecto.

**CNew_Project:** Esta clase es la encargada de controlar el comportamiento de la interfaz UINewProject. Propiamente dicho es la que gestiona los eventos directos del usuario sobre la interfaz de Nuevo Proyecto.

**VPro_Plugin_Worker:** Esta es la clase encargada de cargar dinámicamente la información de los distintos tipos de proyectos que soporta el IDE. Propiamente dicho se encarga de buscar los directorios correspondientes a los plugins y cargar en memoria la información referente a los proyectos asociados a cada uno de estos plugins, información que es utilizada posteriormente por el Asistente de Creación de Proyecto para mostrar dicha información como por ejemplo: nombre de los proyectos, categoría en la que se enmarca cada proyecto, descripción del mismo, imagen o logo asociado al proyecto. Toda esta información es cargada al iniciar el IDE y se realiza en un hilo aparte del principal del Sistema IDE.

### 3.2.4 Clases del Diseño Gestionar Mensajes

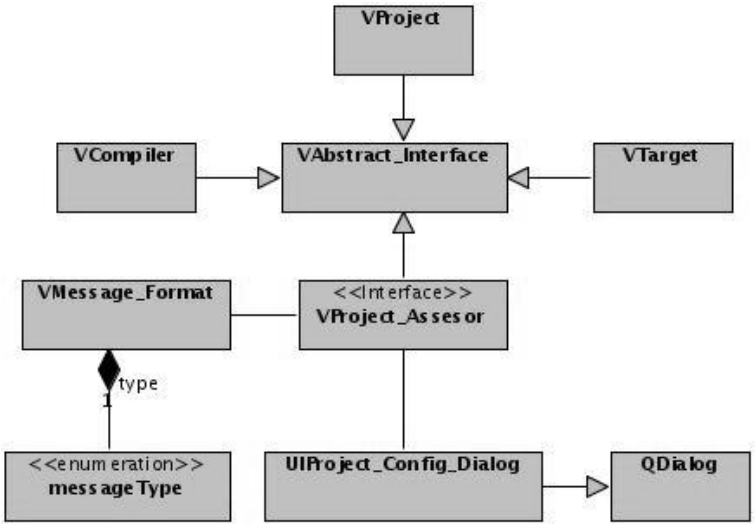



**QDockWidget:** Clase perteneciente a las librerías QT. Esta clase provee una ventana que puede ser incrustada dentro de un QMainWindow o flotar como una ventana independiente en el IDE.

**UIApplication_Message:** Esta clase se encarga de capturar los mensajes emitidos por VSystem referentes a los mensajes que lanza la salida final del proyecto cuando está en ejecución. Es decir, cuando ejecutamos la salida final, la misma puede emitir mensajes tanto por la salida estandar (stdout) como por la salida de error estandar (stderr), tales señales son capturadas por el sistem IDE y mostradas mediante la interfaz UIApplicacion_Message.

**UICompiler_Message:** Se encarga de capturar los mensajes que emite VSystem en relación con los mensajes lanzados por el compilador a la hora de construir la salida final del proyecto. Es decir, cuando se ordena construir el proyecto, el compilador emite diferentes tipos de mensajes, tales como los mensajes de errores y de alertas, tales mensajes son capturados por el sistema IDE y visualizados por UICompiler_Message.

### 3.2.5 Clases del Diseño Sistema de Plugins




**VAbstract_Interface:** Esta clase está hecha con el objetivo de crear una interfaz común a varias clases que no presentan una relación jerárquica específica. Es simplemente para agrupar comportamiento común que presentan varias clases que no pertenecen al mismo árbol de herencia.

**VProject:** Esta clase es la entidad encargada de almacenar los datos comunes a todo proyecto. Es una clase abstracta por lo que no se pueden crear objetos de ella. Para crear una configuración personalizada del proyecto se debe heredar de la misma.

**VCompiler:** Esta clase es la entidad encargada de almacenar los datos comunes de un compilador. Es una clase abstracta por lo que no se pueden crear objetos de ella. Para crear una configuración personalizada del compilador se debe heredar de la misma.

**VTarget:** Esta clase es la entidad encargada de almacenar los datos pertenecientes a un target (salida final del proyecto). Es una clase abstracta por lo que no se crean objetos de la misma. Para implementar un target personalizado se debe heredar de ella.

**MessageType:** Es un ***enum*** que contiene constantes para enmarcar cada tipo de mensaje emitido por compilador. Explícitamente las constantes que presenta son ERROR = 0 , WARNING = 1 , INFO = 2.

**VMessage_Format:** Es una estructura que contiene las especificaciones de formato del mensaje emitido por el compilador, la misma guarda información que puede utilizarse posteriormente para saber

si es un mensaje de error, la línea del mismo, el fichero en el cual se reporta tal error. Esta clase contiene un atributo del tipo MessageType.

**UIProject_Config_Dialog:** Es la clase encargada de construir la interfaz de usuario del Asistente de Configuración de Proyecto, así como gestiona algunos pocos eventos del usuario.

### 3.4 Concepción general de la ayuda

La aplicación contará con un Manual de Usuario, el cual será distribuido junto al software en formato pdf, inicialmente estará en español, pero se traducirá a otros idiomas como el inglés. En estos momentos el documento se encuentra en elaboración, debido a que la prioridad principal ha sido la implementación del sistema, por lo que no será incluida en las primeras versiones.

### 3.5 Estándar de Codificación

Los estándares de codificación son reglas específicas a un lenguaje que reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores. Los estándares de codificación no destapan problemas existentes, evitan más bien que los errores ocurran. Los errores frecuentes en programas pueden ser detectados mucho anterior o pueden incluso ser evitados totalmente. Durante el desarrollo, estándares de codificación ayudan a los ingenieros a producir un código de alta calidad y a entender y a utilizar el código de sus colegas. Pero también realzan considerablemente la capacidad de mantenimiento y reutilización a largo plazo del producto final. Tal práctica del control de errores en el proceso del desarrollo mejora la calidad mientras que reduce el tiempo de desarrollo, el coste, y el esfuerzo.

Para la implementación del sistema se definieron las siguientes pautas de codificación:

- **Idioma**

Todos los nombres de clases, funciones y variables se definen en inglés.

- **Nombres de Clases**
  - Cada palabra comienza con mayúscula.
  - Los nombres de clases que incluyan más de una palabra irán separados por el signo "_ ".
  - Ejemplo: class Project_Manager

- **Tipos de Clases**
  - Clases Interfaz de Usuario: comienzan siempre con el prefijo UI , seguido por el nombre de la clase.
  - Clases Controladoras: comienzan con la letra C, seguida por el nombre de la clase.
  - Clases del Negocio: comienzan con la letra V, seguida por el nombre de la clase.
  - Ejemplos: VProject_Manager, UIMain_Window, CMain_Window .

- **Nombres de Atributos de Clases**
  - Todos los nombres de variables se definen en minúsculas.
  - Deben utilizarse nombres descriptivos que permitan una comprensión rápida de la función de dicha variable.
  - Los nombres de variables que incluyan más de una palabra irán separados por el signo "_ ".

Ejemplo: bool compiler_options[ ] .

- **Nombres de Funciones**
  - Deben utilizarse nombres descriptivos que permitan identificar rápidamente el objetivo de la función, sin importar cuán largo sea el nombre.
  - Los nombres de variables que incluyan más de una palabra irán separados por el signo "_ ".
  - Comienzan con letra minúsculas, y en caso de existir, el resto de las palabras que forman el nombre comenzarán con mayúsculas.

Ejemplo: get_User_List ( )

- **Nombres de Argumentos de Funciones**
  - Comienzan con letra minúsculas, y en caso de existir, el resto de las palabras que forman el nombre comenzarán con mayúsculas.
  - Los nombres de variables que incluyan más de una palabra irán separados por el signo "_ ".
  - En los prototipos de funciones se incluirá el nombre de los argumentos debido a que esto hace más legible aún las etiquetas de completamiento de código a la hora de programar, obteniendo así una mayor comprensión de qué argumentos se requieren.
- **Nombres de Variables Temporales**
  - Se utilizaran nombres lo más cortos posible
  - Se pueden utilizar prefijos que ayuden a entender el significado de la variable como por ejemplo: ***max*** para identificar que contiene un valor máximo, ***str*** para indicar que contiene un valor en modo de cadena de caracteres.

Ejemplo:

```
int max_id;
```

string str_age = “123”;

- **Constantes Globales**
  - Todas las constantes, ya sean atributos de un ***enum*** como definidas mediante la directiva #define deben escribirse completamente en mayúscula.

Ejemplo:

```
enum VStatus
{
        RUNNING,
        BUILDING
};

enum VBuild_Notifications
{
        RUNNING_ ABORTED,
        BUILDING_ABORTED
};

#define MAX 100
```

### 3.6 Arquitectura del sistema

La Arquitectura de un software puede definirse como un grupo de patrones y abstracciones que proporcionan una guía para la construcción de un software. La arquitectura de un programa se selecciona y define de acuerdo a los objetivos y restricciones que el sistema debe tener, algunas se recomiendan para determinadas tecnologías y otras no.

Para el desarrollo del sistema propuesto se estudió la arquitectura basada en **n** capas, debido a que este modelo brinda una serie de ventajas que posibilitan:

- Desarrollos paralelos (en cada capa)
- Aplicaciones más robustas debido al encapsulamiento.
- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica)
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad, sin tener que modificar).

### 3.7 Patrones GRASP

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable. Los patrones GRASP utilizados en el sistema son:

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

| | | |
|---|---|---|
| | | modo independiente y reutilizarlo en otro proyecto? Para determinar el nivel de acoplamiento de clases, son muy buenos los diagramas de colaboración de UML. Uno de los principales síntomas de un mal diseño y alto acoplamiento es una herencia muy profunda. Siempre hay que considerar las ventajas de la delegación respecto de la herencia. |
| High Cohesion - Alta cohesión | ¿Cómo mantener la complejidad manejable? | Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable.<br>Ejemplos de una baja cohesión son clases que hacen demasiadas cosas. En todas las metodologías se considera la refactorización. Uno de los elemento a refactorizar son las clases saturadas de métodos.<br>Ejemplos de buen diseño se producen cuando se crean los denominados “paquetes de servicio” o clases agrupadas por funcionalidades que son fácilmente reutilizables (bien por uso directo o por herencia). |
| Controller - Controlador | ¿Quién debería ser el responsable de gestionar un evento de entrada al sistema? | Asignar una responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase que representa una de las opciones siguientes:<br>1. Representa el sistema |

| | | |
|---|---|---|
| | | global, dispositivo o subsistema.<br>2. Representa un caso de uso en el que tiene lugar el evento del sistema a menudo denominado <nombre del caso de uso> Manejador, <nombre del caso de uso> coordinador, <nombre del caso de uso> Sesión.<br>•Utilice la misma clase controlador para todos los eventos del sistema en el mismo escenario de caso de uso.<br>•Informalmente, una sesión es una instancia de una conversación con un actor. Las sesiones pueden tener cualquier duración, pero se organizan a menudo en función de casos de uso. |

## 3.8 Conclusiones

En este capítulo se obtuvieron los diagramas de clases de diseño de las principales clases a implementar en el sistema, logrando una visión detallada de sus atributos y las relaciones entre ellas. Se elaboró un diagrama de componentes y se definió el estándar de codificación a utilizar, con el objetivo de que el código resultante sea lo más reutilizable y fácil de mantener en un futuro.

A través de este capítulo se realizó la implementación del sistema, cumpliendo así el principal objetivo de este libro: desarrollar una herramienta para la gestión de código para el Ambiente de Desarrollo Integrado propuesto.

# CAPÍTULO 4

## Estudio de Factibilidad

### 4.1 Introducción

En este capítulo se presenta un estudio completo de factibilidad del producto, estimándose el tiempo de desarrollo que se requiere para la implementación del sistema, el esfuerzo humano y los costos económicos, así como los beneficios tangibles e intangibles que se obtendrán una vez desarrollado el producto, para ellos se utiliza el método de estimación por Puntos de Casos de Uso.

### 4.2 Planificación mediante Puntos de Casos de Uso

**Cálculo de Casos de Uso sin ajustar:**

Para calcular los Casos de Uso sin ajustar se utiliza la ecuación:

UUCP = UAW + UUCW

Donde UUCP son los Puntos de Casos de Uso sin ajustar, UAW es el Factor de Peso de los Actores sin ajustar, y UUCW el Factor de Peso de los Casos de Uso sin ajustar. A continuación se calculan estos valores.

**Cálculo del Factor de Peso de los Actores sin ajustar (UAW) :**

Para realizar el cálculo de UAW se les asigna determinado valor a los actores del sistema en dependencia de su clasificación. En este caso, el actor del sistema se clasifica como Complejo, debido a que es una persona que interactúa con el sistema, realizando diversas operaciones sobre la aplicación de escritorio, por lo que se le asigna valor 3.

Utilizando la fórmula

UAW = cantidad de usuario * factor de peso

Queda

UAW = 1 * 3 = 3

**Cálculo de Factor de Peso de Casos de Uso sin ajustar (UUCW)**

Para asignarle un valor al Factor de Peso de un Caso de Uso, es necesario determinar su nivel de complejidad de acuerdo al número de transacciones que posee, para ellos se usan varios rangos y cada uno tiene un valor asociado.

Teniendo en cuenta esto, se procede a la asignación del Factor de Peso correspondiente a cada Caso de Uso del Sistema:

| Caso de Uso | Complejidad | Factor de Peso |
|---|---|---|
| Gestionar Ficheros | Medio | 10 |
| Cerrar Fichero | Simple | 5 |
| Guardar Fichero | Medio | 10 |
| Reconocer Lexer | Simple | 5 |
| Salir del Sistema | Simple | 5 |
| Gestionar Proyecto | Medio | 10 |
| Configurar Proyecto | Medio | 10 |
| Guardar Proyecto | Simple | 5 |
| Construir Proyecto | Medio | 10 |
| Gestionar Mensajes | Simple | 5 |
| Editar Fichero | Simple | 5 |
| Total | | 80 |

UUCW = 80

Aplicando la fórmula mencionada al inicio:

UUCP = UAW + UUCW

UUCP = 3 + 80 = 83

**Cálculo de Puntos de Casos de Uso ajustados**

El segundo paso es calcular los Puntos de Caso de Uso ajustados, para ello se utiliza la fórmula:

UCP = UUCP x TCF x EF
Donde:
UCP: Puntos de Casos de Uso ajustados
UUCP: Puntos de Casos de Uso sin ajustar
TCF: Factor de complejidad técnica
EF: Factor de ambiente

**Factor de Complejidad Técnica (TCF)**

Fórmula para calcular el Factor de Complejidad Técnica:

TCF **=** 0.6 + 0.01 x Σ (Peso i x Valor asignado i)

TCF **=** 0.6 + 0.01 x 49 = 1.09

**Cálculo Factor de Ambiente (EF)**

EF = 1.4 - 0.03 x Σ (Peso i x Valor asignado i)

EF = 1.4 – 0.03 x 18.5 = 0.845

Una vez calculados el EF, TCF y UUCP se procede al cálculo de Puntos de Casos de Uso ajustados
UCP = UUCP x TCF x EF
UCP = 83 x 1.9 x 0.845
UCP = 133.25

**Estimación del Esfuerzo**

Para realizar la estimación del esfuerzo requerido para el desarrollo de la aplicación se contabilizan cuántos factores de los que afectan al Factor de ambiente están por debajo del valor medio (3), para los factores E1 a E6.

Se contabilizan cuántos factores de los que afectan al Factor de ambiente están por encima del valor medio (3), para los factores E7 y E8.

- Si el total es 2 o menos, se utiliza el factor de conversión 20 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 20 horas-hombre.
- Si el total es 3 o 4, se utiliza el factor de conversión 28 horas-hombre/Punto de Casos de Uso, es decir, un Punto de Caso de Uso toma 28 horas-hombre.
- Si el total es mayor o igual que 5, se recomienda efectuar cambios en el proyecto, ya que se considera que el riesgo de fracaso del mismo es demasiado alto.

Se utiliza el Factor de Conversión (CF) de 20 horas-hombre.
El esfuerzo en horas-hombre viene determinado por la ecuación:
E = UCP x CF

Donde
E: es el esfuerzo estimado en horas-hombre
UCP: son los Puntos de Casos de Uso ajustados
CF: es el factor de conversión

Inicialmente se calcula el Esfuerzo de la Implementación, y a partir de ahí es posible obtener los demás.

Esfuerzo (Implementación) = 133.25 x 20 = 2665 horas-hombre
Esfuerzo Total = 2665 * 100/40 = 6662.5 horas-hombre.

Tiempo de Desarrollo = Esfuerzo Total / Total Horas al Mes
Tiempo de Desarrollo = 6662.5 / 240 = 27 meses

Tiempo de Desarrollo por persona = Tiempo de Desarrollo / Cantidad de Personas
Tiempo de Desarrollo por persona = 27 / 2 = 13.5 meses

Costo Total del Proyecto = 225 pesos * 27 meses = 6075 pesos.

| Actividad | Porcentaje | Horas- Hombre |
|---|---|---|
| Análisis | 15 % | 999.37 |
| Diseño | 25 % | 1665.62 |
| Implementación | 40 % | 2662.5 |
| Prueba | 10 % | 666.25 |
| Sobrecarga | 10 % | 666.25 |
| Total | 100 % | 6662.5 |

### 4.3 Beneficios tangibles e intangibles

#### 4.3.1 Beneficios tangibles

Como beneficios tangibles se obtienen:

- Un software alternativo a las herramientas privativas utilizadas, totalmente libre y 100% autóctono.
- Código reutilizable.
- Acceso al código fuente, lo que permite modificarlo y adaptarlo a las necesidad específicas de cualquier proyecto.
- Posibilidad de comercializarlo.

#### 4.3.2 Beneficios Intangibles

Los beneficios intangibles que se obtienen con el uso de la aplicación son:

- Mayor conocimiento de los lenguajes de programación para multimedia como ActionScript 2.0
- Creación de productos multimedia utilizando el paradigma orientado a objetos.

### 4.4 Análisis de costos y beneficios

La solución propuesta requiere de un 1 año y un mes para su implementación, debido al alto nivel de complejidad que posee, con un costo aproximado de 6075 pesos para su desarrollo, lo cual llevado a pesos convertibles representa 243 CUC, completamente resarcible si se comercializa el producto, sin embargo, si se toma en cuenta que una licencia para utilizar Adobe Flash cuesta 699 dólares (Adobe

System Incorporated 2008) , y se calcula el costo que supondría adquirir las licencias para cada estación de trabajo que lo requiera en la universidad, los costos serían prácticamente imposibles para el país, por lo que se puede afirmar que es factible desarrollar el proyecto.

## 4.5 Conclusiones

En el presente capítulo se realizó un estudio completo sobre la factibilidad de desarrollo del producto. Se estimó el esfuerzo total requerido y los costos aproximados para su desarrollo. También se mencionan los beneficios tangibles e intangibles que se obtendrán, y a para concluir se muestra una tabla con los principales datos de la estimación:

| Parámetros | Valores |
|---|---|
| Esfuerzo Total | 6662 horas - hombre |
| Tiempo de Desarrollo | 13.5 meses |
| Cantidad de Hombres | 2 |
| Salario | 225 Pesos |
| Costo Total en Pesos (MN) | 6075 Pesos |
| Costo Total en Pesos Convertibles (CUC) | 243 CUC |

Con la finalización de este capítulo se logra demostrar que es factible el desarrollo de la solución propuesta.

## CONCLUSIONES

Con la finalización del presente trabajo se cumplieron varios objetivos y metas fundamentales:

- Se realizó un estudio profundo del estado del arte de las herramientas libres existentes en el mundo del software libre para la producción de software educativo multimedia, seleccionándose aquellas que podrían utilizarse como parte de la solución.
- Se obtuvo una lista de requerimientos funcionales y no funcionales que definieron el conjunto de funcionalidades que tendría la aplicación final, así como su comportamiento.
- Con la aplicación del RUP como metodología para el desarrollo se ampliaron los conocimientos de Ingeniería de Software y se realizó el modelado del sistema, obteniéndose una arquitectura sólida y flexible.
- Se investigaron las librerías gráficas y frameworks más utilizados para el desarrollo de aplicaciones con interfaz gráfica de usuario, y se escogió una de ellas, argumentando el por qué.
- La evaluación de varios lenguajes para la programación permitió seleccionar el más adecuado de acuerdo a los objetivos y requerimientos de la aplicación.
- Se cumplió con el objetivo principal de implementar un software multiplataforma para el desarrollo de multimedia usando software libre, el cual fue programado usando C++ como lenguaje para la implementación, Qt4 para la interfaz visual, Visual Paradigm como herramienta CASE, y KDevelop como IDE para el desarrollo.

El sistema desarrollado constituye la primera versión de lo que debe ser una completa alternativa a los software propietarios utilizados actualmente en la UCI, brinda todas las funcionalidades requeridas para desarrollar un producto multimedia, sin embargo es posible añadirle nuevas funcionalidades que permitan mejorar y ampliar el producto, por lo que se espera que se continúe con su desarrollo como proyecto productivo en el Área de Investigación y Producción de la Universidad.

## RECOMENDACIONES

El producto obtenido es una versión inicial de lo que debe ser un completo entorno de desarrollo de aplicaciones multimedia y software educativo, por lo que cuenta solo con una parte de todas las funcionalidades que debe tener, el equipo de desarrollo se centró en implementar aquellas funcionalidades básicas y críticas de un IDE para la gestión de código.

Es por ello que se recomienda:

- Continuar el desarrollo del producto, agregándole nuevas funcionalidades y mejorar las ya implementadas.
- Integrar la aplicación con el gestor visual y con el conjunto de clases y ejercicios reutilizables, para una mayor facilidad de desarrollo de multimedia.
- Desarrollar plugins para nuevos compiladores multimedia como el reciente Flex de Adobe.
- Internacionalizar la aplicación, traduciéndola a otros idiomas.
- Elaborar la documentación de usuario, manuales de ayuda y tutoriales que faciliten el aprendizaje de la herramienta.
- Utilizar el software en la producción de software educativo multimedia en la Universidad.
- Crear una comunidad de desarrollo y usuarios entorno al sistema, permitiendo una mayor retroalimentación sobre errores a corregir y nuevas funcionalidades a agregar.
- Mejorar el completamiento de código brindado por la aplicación.

## BIBLIOGRAFÍA

Adobe System Incorporated, 2008. Adobe - Flash CS3 Professional, Interactive Multimedia, Interactive Design. *Adobe - FlashCS3 Professional*. Disponible en: http://www.adobe.com/products/flash/?promoid=BPDEE.

Alessandro Crugnola, 2006. SE|PY, the editor, the blog. *SEPY, the editor, the blog*. Disponible en: http://www.sepy.it/.

Choike, sociedad civil del sur, 2007. Choike - TICs para el desarrollo: un nuevo enfoque a partir de los Objetivos de Desarrollo del Milenio. Disponible en: http://www.choike.org/nuevo/informes/2945.html

Daniel Fischer, 2008. swfmill swf2xml and xml2swf. *swfmill swf2xml and xml2swf*. Disponible en: http://swfmill.org/.

Florian Delizy, 2007. UIRA » Unfreeze's. *Uira Unfreeze's*. Disponible en: http://www.unfreeze.net/?page_id=52.

Free Software Foundation, 2008. La Definición de Software Libre - Proyecto GNU - Fundación para el Software Libre (FSF). *Fundación para el Software Libre (FSF)*. Disponible en: http://www.gnu.org/philosophy/free-sw.es.

Geetanjali Arora, Balasubramaniam Aiaswamy & Nitin Pandey, 2002. *Programación en C#*, España: Anaya Multimedia.

Herbert Schildt, 1995. *C++ Guia de Autoenseñanza*, Osborne/McGraw-Hill.

Ivar Jacobson, Grady Booch & James Rumbaugh, 2000. *El Proceso Unificado de Desarrollo de Software*, Addison Wesley.

Jasmin Planchette & Mark Summerfield, 2006. *C++ Gui Programming with Qt4*, Prentice Hall.

Jeffrey Savit, Sean Wilcox & Sean Wilcox
Bhuvana Jayaraman, 1999. *Java para la empresa*, McGraw-Hill.

Joseph Schmuller, 1999. *UML en 24 horas*, Prentice Hall.

Julian Smart, Kevin Hock & Stefan CSomor, 2006. *Cross Plataform GUI Programming with wxWidgets* 1º ed., Prentice Hall.

Miguel Ángel De Blas Búrdalo, 2008. KDevelop - un Entorno de Desarrollo Integrado - Página de inicio. Disponible en: http://www.kdevelop.org/ .

Mika Palmu, 2008. FlashDevelop Open Source Flash. *FlashDevelop Open Source Flash*. Disponible en: http://osflash.org/flashdevelop .

Motion Tween, 2008. haXe - ¡Bienvenido a haXe! *haxe*. Disponible en: http://haxe.org/.

Motion Tween, 2007. Motion-Twin. *Motion Tween*. Disponible en: http://www.mtasc.org/.

Rainer Böhme & Matthias Kramm, 2007. SWFTOOLS. *SWFTOOLS*. Disponible en: http://www.swftools.org/about.html.

Sergi Perez Maña, 2007. Qflash - flash maker - website. *QFlash -flash maker - website*. Disponible en: http://qflash.sourceforge.net/webpage/.

f4l Team, 2005. Flash For Linux | Free flash desing tools for GNU/Linux. *Flash For Linux | Free flash design tools for GNU/Linux*. Disponible en: http://f4l.sourceforge.net/.

The Code::Blocks Team, 2008. CodeBlocks. *CodeBlocks Home*. Disponible en: http://www.codeblocks.org/.

The Eclipse Foundation, 2008. Eclipse.org home. Disponible en: http://www.eclipse.org/.

The Gtk Team, 2008. GTK+ - About. *GTK*. Disponible en: http://www.gtk.org/.

Tomka Films, 2008. KToon: Herramienta de Animación en 2D - Home. *KToon, Herramienta de Animación en 2D*. Disponible en: http://ktoon-es.toonka.com/index.php?option=com_frontpage&Itemid=1

## GLOSARIO DE TÉRMINOS

**A**

**ActionScript:** lenguaje de scripts muy parecido al JavaScript, pero diseñado específicamente para la creación de multimedias interactivas utilizando Flash. Comenzó por la versión 1.0 y debido al enorme éxito se encuentra en la versión 3.0, es Orientado a Objetos semejante a los lenguajes de programación tradicionales como C++ y Java.

**Applet:** un applet es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El applet debe ejecutarse en un contenedor, que lo proporciona un programa anfitrión, mediante un plugin, o en aplicaciones como teléfonos móviles que soportan el modelo de programación por applets.
A diferencia de un programa, un applet no puede ejecutarse de manera independiente, ofrece información gráfica y a veces interactúa con el usuario, típicamente carece de sesión y tiene privilegios de seguridad restringidos. Un applet normalmente lleva a cabo una función muy específica que carece de uso independiente. El término fue introducido en AppleScript en 1993.
Ejemplos comunes de applets son las Java applets y las animaciones Flash. Otro ejemplo es el Windows Media Player utilizado para desplegar archivos de video incrustados en los navegadores como el Internet Explorer.

**Automake:** GNU Automake es una herramienta de programación que produce programas makefiles portables para el uso de make usado en la compilación de software. Es mantenido por la Fundación de Software Libre como uno de los programas GNU y es parte del sistema de construcción GNU. El archivo Makefile generado sigue la directiva Estándar de Codificación GNU.

**B**

**Bindings:** en lenguajes de programación es la referencia al identificador de un valor, también se refiere al intercambio de algunas funciones en determinados contextos. Ejemplo, cuando se dice que una librería está escrita en Java y tiene bindings para C++ significa que provee una interfaz para este lenguaje, permitiendo su uso sin modificarla.

**Bytecode:** el bytecode es un código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto,

que es un fichero binario producido por el compilador cuyo contenido es el código objeto o código máquina.
Los programas en bytecode suelen ser interpretados por un intérprete (en general llamado máquina virtual, dado que es análogo a un ordenador). Su ventaja es su portabilidad: el mismo código binario puede ser ejecutado en diferentes plataformas y arquitecturas.

**C**

**Compilador:** es un programa modo consola que permite traducir fichero código fuente de un lenguaje de programación determinado a lenguaje máquina, generando así un ejecutable.

**CVS:** el Concurrent Versions System (CVS), también conocido como Concurrent Versioning System, es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto (de programa) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren. CVS se ha hecho popular en el mundo del software libre. Sus desarrolladores difunden el sistema bajo la licencia GPL.

**D**

**Debugger:** un depurador (en inglés, debugger), es un programa que permite depurar o limpiar los errores de otro programa informático. Depuración de programas es el proceso de identificar y corregir errores de programación. En inglés se le conoce como debugging, ya que se asemeja a la eliminación de bichos (bugs), manera en que se conoce informalmente a los errores de programación.

**Doxygen:** es un Generador de documentación para C++, C, Java, Objective-C, Python, IDL (versiones Corba y Microsoft) y en cierta medida para PHP, C# y D. Dado que es fácilmente adaptable, funciona en la mayoría de sistemas Unix así como en Windows y Mac OS X.
Doxygen es un acrónimo de dox(document) gen(generator), generador de documentación para código fuente.

**E**

**Editor de Código:** es un programa que permite editar código escrito para determinado lenguaje, generalmente ejecuta el coloreado de sintaxis de las palabras claves del lenguaje, sin embargo no cuenta con herramientas para la compilación y construcción de proyectos.

**enum:** es un recurso utilizado en C++ para definir variables constante de la siguiente forma :
enum combustible {GASOLINA = 0, PETROLEO = 1};
De esta forma quedan definidas dos constantes enteras GASOLINA y PETROLEO cada una con los valores o y 1 respectivamente. También enum se puede utilizar como tipo de dato, por lo que se pueden declarar variables del tipo de dato combustible.

**F**

**Framework:** Es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo. Son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional.
Fuera de las aplicaciones en la informática, un framework puede ser considerado como el conjunto de procesos y tecnologías usados para resolver un problema complejo. Es el esqueleto sobre el cual varios objetos son integrados para una solución dada.

**Frontend:** front-end y back-end son términos que se relacionan con el principio y el final de un proceso. El front-end es la parte del software que interactúa con el o los usuarios y el back-end es la parte que procesa la entrada desde el front-end. La separación del sistema en "front ends" y "back ends" es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas. La idea general es que el front-end sea el responsable de recolectar los datos de entrada del usuario, que pueden ser de muchas y variadas formas, y procesarlas de una manera conforme a la especificación que el back-end pueda usar. La conexión del front-end y el back-end es un tipo de interfaz.

**G**

**GNU/Linux:** sistema operativo libre y gratuito basado en Unix, el cual basa su desarrollo en la comunidad que lo sustenta, aunque existen varias empresas que lo distribuyen como Red Hat y Novell. Una de sus principales características es que el código fuente está disponible, siendo licenciados bajo GPL.

**H**

**Herramienta CASE:** son programas diseñados para el modelado de sistemas, generalmente soportan varias metodologías de desarrollo como RUP, XP, MSF. Ejemplo de estas aplicaciones son Umbrello, Visual Paradigm, Rational Rose.

**I**

**IDE:** Ambiente de Desarrollo Integrado (Integrated Development Enviroment). Un IDE es un programa que integra un conjunto de herramientas para el desarrollo de software, automatizando las tareas que repite a menudo como compilar, construir, ejecutar, creación de nuevos proyectos, configuración de opciones, entre otras. Además brindan facilidades al proceso de edición de código resaltando las palabras resaltadas del lenguaje y permitir el autocompletamiento.

**J**

**JVM:** Java Virtual Machine (Máquina Virtual de Java) es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java.

**L**

**Licencias de software:** especie de contrato donde se especifican las normas y reglas para utilizar determinado software, entre estas normas están los permisos de copia, distribución y modificación del programa. Existen muchos tipos de licencias, sin embargo se dividen generalmente en dos grupos: privativas y libres.

**Linux:** es un sistema operativo tipo Unix (también conocido como GNU/Linux) que se distribuye bajo la Licencia Pública General de GNU (GNU GPL), es decir que es software libre. Su nombre proviene del Núcleo de Linux, desarrollado en 1991 por Linus Torvalds. Las variantes de estos sistemas se denominan "distribuciones" y su objetivo es ofrecer una edición que cumpla con las necesidades de determinado grupo de usuarios. De esta forma existen distribuciones para hogares, empresas y servidores. Algunas son gratuitas y otras de pago, algunas insertan software no libre y otras contienen solo software libre.

**M**

**Mac OSX:** (pronunciado Mac-o-ese-diez) es el actual sistema operativo de la familia de ordenadores Macintosh. Mac OS X es un sistema operativo basado en UNIX, pero donde el gestor de ventanas X11, característico de estos sistemas, ha sido sustituido por otro denominado Aqua, desarrollado íntegramente por Apple.

**Makefile:** El archivo de texto que sigue la directiva Estándar de Codificación GNU usado por la herramienta GNU Automake.

**MovieClip:** concepto manejado por Flash para definir cierto objeto que tiene un comportamiento parecido a una película de cine, el cual presenta una secuencia de fotogramas cada determinado intervalo.

**Multimedia:** Aplicación que integra varios recursos comunicativos y audiovisuales para comunicar determinado conocimiento, y permite al usuario interactuar con el contenido del mismo.

**O**

**Operador Unario:** Los operadores unarios realizan una operación en un solo operando, como devolver el valor positivo o negativo de una expresión numérica. Ejemplo de operadores unarios +, - , *, /.

**Open Source:** Traducido al español open (abierto), source (fuente). En programación este término se utiliza para definir a aquel software o código al cual se puede acceder a sus fuentes, llámese fuente a los ficheros de código y todos los recursos que se usaron para la creación de dicho software.

**P**

**Perl:** lenguaje Práctico para la Extracción e Informe; es un lenguaje de programación diseñado por Larry Wall creado en 1987. Toma características del lenguaje C, del lenguaje interpretado shell (sh), AWK, sed, Lisp y, en un grado inferior, de muchos otros lenguajes de programación.

**Plugin:** un plugin o componente enchufable (o plug-in -en inglés "enchufar"-, también conocido como addin, add-in, addon o add-on) es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica, generalmente muy específica, como por ejemplo servir como driver (controlador) en una aplicación, para hacer así funcionar un dispositivo en otro programa. Actualmente como una forma de expandir programas de forma modular, de manera que se puedan añadir nuevas funcionalidades sin afectar a las ya existentes ni complicar el desarrollo del programa principal.

**Python:** lenguaje de Programación interpretado creado por Guido van Rossum en el año 1990.

**Q**

**Qmake:** Es una herramienta del framework de Qt que ayuda a simplificar el proceso de desarrollo de proyectos a través de diferentes plataformas. Esta herramienta automatiza la creación de los ficheros Makefiles por lo que solamente pocas líneas son necesarias para construir cada Makefile. Qmake puede ser usado para cualquier tipo de proyecto escrito en Qt o no.

**Qt Designer:** Herramienta perteneciente al framework de Qt que permite al desarrollador diseñar las interfaces de usuario de su programa.

**R**

**RUP: Rational** Unified Process (Proceso Unificado de Desarrollo de Software). Es una metodología estándar que junto al UML se utiliza para el desarrollo de software.

**S**

**Señales y Slots:** estos términos son acatados por las librerías Qt para referirse a la comunicación entre objetos. El mecanismo de Señales y Slots es una de las características centrales de Qt y probablemente la parte que difiere de las características que proveen la mayoría de otros frameworks. Una Señal es emitida por una clase cuando ocurre algún evento particular. Un Slot es una función que es llamada en respuesta a una señal determinada.

**Software Libre:** se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. De modo más preciso, se refiere a cuatro libertades fundamentales de los usuarios del software planteadas por la Fundación de Software Libre (FSF), las cuales son: libertad de usar el programa para cualquier propósito, libertad de estudiar como funciona el programa y adaptarlo a sus necesidades, la libertad de distribuirlo, la libertad de mejorar el programa y hacer públicas las mejoras a los demás, logrando un beneficio a la comunidad.

**Software Privativo:** dícese de todo aquel software que no cumple con alguna de las 4 libertades fundamentales planteadas por la FSF, entre ellos se encuentran los Contratos de Licencia de Acuerdo Final del Usuario, que limitan enormemente los derechos de un usuario.

**T**

**TICs :** se definen como un conjunto de tecnologías que permiten el acceso a la información y la comunicación de una forma rápida y eficaz, involucran Internet, Correo Electrónico, Televisión Interactiva, Aprendiza a través de medios electrónicos, entre otros.

**U**

**UIC:** herramienta perteneciente al framework de Qt para la generación de código C++ y Qt a partir de un fichero en formato XML .ui que representa la información de la interfaz de usuario a construir.

**Unix:** es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

**W**

**Widgets:** es un término que acatan las librerías Qt para referirse a todo componente de interfaz de usuario como botones, ventanas, cajas de texto, debido a que cada uno tiene la capacidad de mostrarse individualmente en una ventana flotante si no le es definido otro widget en el cual incrustarse.

# ANEXOS

## ANEXO A. Descripción de los Casos de Uso del Sistema.

| | |
|---|---|
| **Caso de Uso:** | **Gestionar Ficheros** |
| **Actores:** | **Desarrollador** |
| **Resumen:** | El caso de uso se ejecuta cuando el desarrollador decide crear un nuevo fichero, abrir un fichero existente, cerrar un fichero, o cerrar todos los ficheros abiertos. |
| **Referencia:** | **R3, R7** |
| **CU asociados:** | CU Guardar Fichero CU Reconocer Lexer |
| **Precondiciones:** | |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El desarrollador selecciona el Menú Archivo. | 1. El sistema despliega el menú Archivo con las siguientes opciones:<br>**a.** Abrir Archivo ( Sección **Abrir Fichero**)<br>**b.** Crear Nuevo Archivo ( Sección **Crear Fichero** ) |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | |
| **Poscondiciones:** | |
| **Prioridad:** | **Normal** |
| **Especificaciones Complementaria:** | |
| **Sección Abrir Fichero** | |
| **Flujo normal de Eventos** | |
| **Acción del Actor** | **Respuesta del sistema** |

| | |
|---|---|
| **1.** El desarrollador selecciona la opción Abrir Archivo. | **1.1** El sistema muestra una ventana que le permite al usuario seleccionar la ruta al archivo que desea abrir. |
| **2.** El desarrollador selecciona el archivo que desea abrir. | **2.1** El sistema carga el archivo y muestra su contenido.<br>**2.2** El sistema invoca el Caso de Uso incluido **Reconocer Lexer.** |
| **Sección Crear Fichero** | |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El usuario selecciona la opción Crear Nuevo Archivo en el menú Archivo. | **1.1** El sistema muestra una ventana con los tipos de archivo que soporta, además permite especificar el nombre del archivo y la ruta a donde será creado. |
| **2.** El usuario selecciona el tipo de archivo que va a crear, especifica el nombre del mismo y la ruta donde se creará | **2.1** El sistema crea el fichero en la ruta especificada.<br>**2.2** El sistema invoca el Caso de Uso incluido **Reconocer Lexer.**<br>**2.3** El sistema muestra el contenido del archivo creado. |

**Tabla A 1 Descripción de Casos de Uso Gestionar Ficheros.**

| | |
|---|---|
| **Caso de Uso:** | **Cerrar Fichero** |
| **Actores:** | **Desarrollador** |
| **Resumen:** | El caso de uso se ejecuta cuando el desarrollador decide cerrar un fichero. |
| **Referencia:** | **R8, R9** |
| **CU asociados:** | CU Guardar Fichero |
| **Precondiciones:** | |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El desarrollador selecciona la opción Cerrar Archivo. | **1.1** El sistema verifica que el contenido del archivo no haya sido modificado desde la última vez que fue guardado. Si fue |

| | |
|---|---|
| | modificado se ejecuta el Caso de Uso **Guardar Fichero.**<br>**1.2** El sistema cierra el archivo, y activa el siguiente fichero que se encuentre abierto. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | **1.3** En caso de no existir ningún fichero abierto, el sistema desactivará las opciones de Guardar, Guardar como, Cerrar y Cerrar Todo. |
| **Poscondiciones:** | |
| **Prioridad:** | **Normal** |
| **Especificaciones Complementaria:** | |

**Tabla A 2 Descripción de Casos de Uso Cerrar Fichero**

| | |
|---|---|
| **Caso de Uso:** | **Guardar Fichero** |
| **Actores:** | **Desarrollador** |
| **Resumen:** | El Caso de Uso se ejecuta cuando el usuario selecciona alguna de las opciones para guardar el contenido del fichero que se encuentra editando. Estas opciones pueden ser: Guardar, Guardar como, y Guardar todo. |
| **Referencia:** | **R4, R5, R6** |
| **CU asociados:** | |
| **Precondiciones:** | Debe estar abierto algún fichero. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El usuario selecciona la opción **Guardar** en el Menú Archivo. | **1.1** El sistema guarda el contenido del fichero en el disco. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | **1.2** Si es la primera vez que se guarda el archivo se ejecuta la Sección **Guardar como**. |

| **Prioridad:** | **Crítico** |
|---|---|
| **Especificaciones Complementaria:** | |
| **Sección Guardar como** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El usuario selecciona en el menú Archivo la opción **Guardar como** | **1.1** El sistema muestra una ventana de diálogo donde el usuario selecciona una ruta y un nombre para guardar el archivo. |
| **2.** El usuario inserta la nueva ruta y el nombre del usuario. | **2.1** El sistema guarda el archivo bajo el nombre y ruta especificada. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **2.** El usuario selecciona la opción Cancelar en la Ventana de diálogo mostrada por el sistema. | **2.1** El sistema oculta la ventana y no se guarda el archivo. |
| **Sección Guardar todo** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El usuario selecciona en el menú Archivo la opción **Guardar todo** | **1.1** El sistema guardar en disco todos los ficheros que se encuentren abiertos en ese momento. |
| **Flujos alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | **1.2** El sistema encuentra un archivo que no ha sido guardado por primera vez. Se ejecuta la Sección **Guardar como.** |

**Tabla A 3 Descripción de Casos de Uso Guardar Fichero**

| **Caso de Uso:** | ***Reconocer Lexer*** |
|---|---|
| **Actores:** | **Desarrollador** |
| **Resumen:** | El Caso de uso es invocado por los casos de uso **Crear Fichero** o **Abrir Fichero,** y permite realizar el coloreado y resaltado de las palabras claves del lenguaje al que pertenece el archivo |
| **Referencia:** | **R10** |
| **CU asociados:** | **Gestionar Fichero** |

| | |
|---|---|
| **Precondiciones:** | |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** | **1.1** El sistema identifica la extensión del archivo que se encuentra en edición.<br>**1.2** El sistema solicita al Gestor de Plugins el lexer correspondiente a dicho tipo de archivo.<br>**1.3** El sistema recibe el lexer, luego busca en todo el texto del archivo en edición las palabras claves y las colorea de acuerdo a un conjunto de reglas definidas en el lexer. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | |
| **Poscondiciones:** | |
| **Prioridad:** | **Crítico** |
| **Especificaciones Complementaria:** | |

**Tabla A 4 Descripción de Casos de Uso Reconocer Lexer**

| | |
|---|---|
| **Caso de Uso:** | **Salir del Sistema** |
| **Actores:** | **Desarrollador** |
| **Resumen:** | El Caso de Uso se inicia cuando el desarrollador selecciona la opción Salir, tanto en el Menú Archivo o presionando el botón Salir de la ventana principal. |
| **Referencia:** | **R27** |
| **CU asociados:** | **Cerrar fichero** |
| **Precondiciones:** | |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El desarrollador selecciona la opción Salir | **1.1** El sistema verifica si ha ocurrido algún cambio en los archivos que se encuentren abiertos luego de la última vez en que |

| | |
|---|---|
| | fueron guardados.<br>**1.2** El sistema invoca el Caso de Uso **Cerrar Fichero.**<br>**1.3** El sistema finaliza todos los procesos abiertos y termina su ejecución. |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | |
| **Poscondiciones:** | |
| **Prioridad:** | **Crítico** |

**Tabla A 5 Descripción de Casos de Uso Salir del Sistema**

| | |
|---|---|
| **Caso de Uso:** | **Gestionar Proyecto** |
| **Actores:** | **Desarrollador** |
| **Resumen:** | Este Caso de Uso se inicia cuando el desarrollador decide crear un nuevo proyecto, abrir un proyecto existente o cerrar el que esta editando.<br>Si el desarrollador presenta un proyecto abierto puede a su vez configurar las opciones del proyecto, del compilador y de la salida que se va a generar. |
| **Referencia:** | **R11, R16, R20** |
| **CU asociados:** | **Gestionar Archivo de Configuración. Gestionar Ficheros** |
| **Precondiciones:** | |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El desarrollador selecciona el menú Proyecto. | **1.** El sistema despliega el menú Proyecto con las distintas opciones:<br>**1.1** Nuevo Proyecto (sección **Nuevo Proyecto**).<br>**1.2** Abrir Proyecto (sección **Abrir Proyecto**).<br>**1.3** Cerrar Proyecto (sección **Cerrar Proyecto**).<br>**1.4** Configuración del Proyecto (Caso de Uso Configurar Proyecto). |
| **Flujos Alternos** | |
| **Acción del Actor** | **Respuesta del Sistema** |

| | |
|---|---|
| **Sección Nuevo Proyecto** | |
| **Acciones del Actor** | **Respuestas de Sistema** |
| **1.** El desarrollador selecciona la opción crear Nuevo Proyecto en el menú Proyecto | **1.1** El sistema solicita al Gestor de Plugins información sobre los proyectos que pueden crearse, los cuales están definidos en los plugins de proyecto.<br>**1.2** El sistema muestra una ventana de diálogo en la que están representados los distintos tipos de proyecto que se pueden crear, así como también la posibilidad de seleccionar la ruta en la cual se va a crear el proyecto y el nombre bajo el cual se va guardar el mismo. |
| **2.** El desarrollador selecciona el tipo de proyecto, la ruta y el nombre, y crea el proyecto. | 2.1 El sistema carga el Plugin de acuerdo al tipo de proyecto seleccionado<br>**2.2** El sistema carga la información correspondiente al proyecto seleccionado.<br>**2.2** El sistema crea las carpetas y los archivos correspondientes al proyecto en el directorio especificado.<br>**2.3** Se inicia la Sección Abrir Archivo del Caso de Uso Gestionar Ficheros abriendo el fichero fuente principal del programa.<br>a) **2.4** El sistema activa las opciones de los Casos de Uso:<br>a) Gestionar Archivo Configuración.<br>b) Construir Proyecto. |
| **Sección Abrir Proyecto** | |
| **Acciones del Actor** | **Respuestas del Sistema** |
| **1.** El desarrollador selecciona la opción Abrir Proyecto del Menú Proyecto. | **1.1** El sistema muestra una ventana de dialogo para señalar la ruta del proyecto a abrir. |
| **2.** El desarrollador busca la ruta del proyecto. | **2.1** El sistema carga toda la información necesaria del proyecto.<br>**2.** **2.2** El sistema carga el Plugin asociado al proyecto. |

| | |
|---|---|
| | **2.2** El sistema activa las opciones de los Casos de Uso:<br>a) Gestionar Archivo Configuración.<br>b) Construir Proyecto. |
| **Sección Cerrar Proyecto** | |
| **Acciones del Actor** | **Respuestas del Sistema** |
| **1.** El desarrollador selecciona la opción Cerrar Proyecto del Menú Proyecto. | **1.1** El sistema ejecuta la Sección Cerrar Todo del Caso de Uso Cerrar Archivo.<br>**1.2** El sistema desactiva las opciones de los Casos de Uso:<br>a) Gestionar Archivo Configuración.<br>b) Construir Proyecto. |

**Tabla A 6 Descripción de Casos de Uso Gestionar Proyecto**

| **Caso de Uso:** | **Configurar Proyecto** |
|---|---|
| **Actores:** | **Desarrollador** |
| **Resumen:** | El caso de Uso de inicia cuando el desarrollador decide cambiar las opciones del proyecto. |
| **Referencia:** | **R12, R17, R18, R19, R23, R24** |
| **CU asociados:** | **Guardar Proyecto** |
| **Precondiciones:** | Debe existir un proyecto en edición. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El desarrollador selecciona las opciones Configuración del Proyecto del Menú Proyecto. | **1.1** El sistema muestra una ventana de diálogo con las siguientes opciones:<br>**1.** Opciones de Proyecto<br>**2.** Opciones de la Salida<br>**3.** Opciones del Compilador<br>**4.** Editar Comandos para Ejecutar<br>**5.** Agregar Comandos para Construcción |
| **2.** El desarrollador modifica las opciones que desea, y acepta los cambios. | **2. 2.1** El sistema ejecuta el Caso de Uso Guardar Proyecto. |
| **Flujos Alternos** | |
| **Acciones del Actor** | **Respuestas del Sistema** |

| | |
|---|---|
| **2.** El desarrollador cancela los cambios de configuración. | **2.1** El sistema cierra la ventana de Configuración del Proyecto sin Guardar los cambios. |

**Tabla A 7 Descripción de Casos de Uso Configurar Proyecto**

| **Caso de Uso:** | **Guardar Proyecto** |
|---|---|
| **Actores:** | **Desarrollador** |
| **Resumen:** | El caso de Uso de inicia cuando el desarrollador una vez cambiado las configuraciones del Proyecto acepta dichos cambios. |
| **Referencia:** | **R13, R14, R15** |
| **CU asociados:** | |
| **Precondiciones:** | Debe existir un proyecto en edición. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El desarrollador acepta las modificaciones realizadas en la configuración del proyecto haciendo click en el botón Aceptar de la ventana Configuración de Proyecto, y son guardadas las opciones del proyecto. | **1.1** El sistema guarda en el fichero de configuración del Proyecto las opciones del proyecto abierto.<br>**3. 1.2** El sistema guarda en el fichero de configuración del Proyecto las opciones del compilador asociado al proyecto abierto.<br>**4. 1.3** El sistema guarda en el fichero de configuración del Proyecto las opciones de la salida final definida en el proyecto.<br>**5. 1.4** El sistema guarda en el fichero de configuración del Proyecto los comandos para ejecutar la salida final del Proyecto.<br>**6. 1.5** El sistema guarda en el fichero de configuración del Proyecto los comandos a agregar a la construcción de la salida final. |

**Tabla A 8 Descripción de Casos de Uso Guardar Proyecto**

| **Caso de Uso:** | **Construir Proyecto.** |
|---|---|
| **Actores:** | **Desarrollador** |
| **Resumen:** | El caso de Uso de inicia cuando el desarrollador decide construir y ejecutar el Proyecto.<br>Construir el proyecto significa compilar todos los archivos fuentes del proyecto y generar a partir de estos un ejecutable que sería la aplicación final. |
| **Referencia:** | **R21, R22** |
| **CU asociados:** | **Gestionar Mensajes** |
| **Precondiciones:** | Debe existir un proyecto en edición. |

| **Flujo Normal de Eventos** | |
|---|---|
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El desarrollador selecciona la opción Construir en el Menú. | **1.1** El sistema muestra un menú con las opciones:<br>1) Construir Proyecto (Sección Construir Proyecto).<br>2) Ejecutar Proyecto (Sección Ejecutar Proyecto). |
| **Sección Construir Proyecto** | |
| **Acciones del Actor** | **Respuestas del Sistema** |
| **2.** El desarrollador selecciona la opción Construir Proyecto del Menú Principal Construir. | **2.1** El sistema elabora la línea de comando a pasar al compilador.<br>**2.2** El sistema ejecuta el compilador asociado al proyecto, pasándole la línea de comando correspondiente.<br>**2.3** Se ejecuta la Sección Mostrar Mensajes de Construcción del Caso de Uso Gestionar Mensajes. |
| **Sección Ejecutar Proyecto** | |
| **Acciones del Actor** | **Respuestas del Sistema** |
| **2.** El desarrollador selecciona la opción Ejecutar Proyecto del Menú Principal Construir. | **2.1** El sistema ejecuta la salida final.<br>**2.2** Se inicia la Sección Mostrar Mensajes de Ejecución del Caso de Uso Gestionar Mensajes. |

**Tabla A 9 Descripción de Casos de Uso Construir Proyecto**

| **Caso de Uso:** | **Gestionar Mensajes** |
|---|---|
| **Actores:** | **Desarrollador** |
| **Resumen:** | El caso de uso es iniciado por el Caso de Uso Construir Proyecto de inicia cuando el desarrollador decide construir y ejecutar el Proyecto. |
| **Referencia:** | **R24, R25** |
| **CU asociados:** | |
| **Precondiciones:** | Debe existir un proyecto en edición. |
| **Flujo Normal de Eventos** | |
| **Sección Mostrar Mensajes de Construcción** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | **1.1** El sistema muestra mensajes propios del mismo como son por ejemplo estado de la compilación o construcción del proyecto, así como los comandos a ejecutar por el compilador.<br>**7. 1.2** El sistema captura y muestra los mensajes emitidos por el compilador, de errores, alertas y otros tipos de mensajes propios del mismo. |
| **Sección Mostrar Mensajes de Ejecución** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| | **2.1** El sistema muestra mensajes propios como por ejemplo los pasos a seguir para la ejecución y los comandos a ejecutar.<br>**2.2** El sistema una vez ejecutada la salida final del proyecto captura y muestra los mensajes que emite dicha salida en tiempo de ejecución. |

**Tabla A 10 Descripción de Casos de Uso Gestionar Mensaje**

| **Caso de Uso:** | **Editar Fichero** |
|---|---|
| **Actores:** | **Desarrollador** |
| **Resumen:** | El Caso de Uso procede cuando el usuario introduce texto en el fichero abierto. |
| **Referencia:** | **R10, R25** |
| **CU asociados:** | **Reconocer Lexe**r |
| **Precondiciones:** | Debe encontrarse abierto al menos un archivo. |
| **Flujo Normal de Eventos** | |
| **Acción del Actor** | **Respuesta del Sistema** |
| **1.** El desarrollador introduce texto en el fichero abierto. | **1.1** El sistema invoca el Caso de Uso incluido Reconocer Lexer.<br>**1.2** El sistema realiza el coloreado de las palabras claves existentes en el texto del fichero. |

**Tabla A 11 12 Descripción de Casos de Uso Editar Fichero**

## ANEXO B. Diagrama de Clases del Diseño

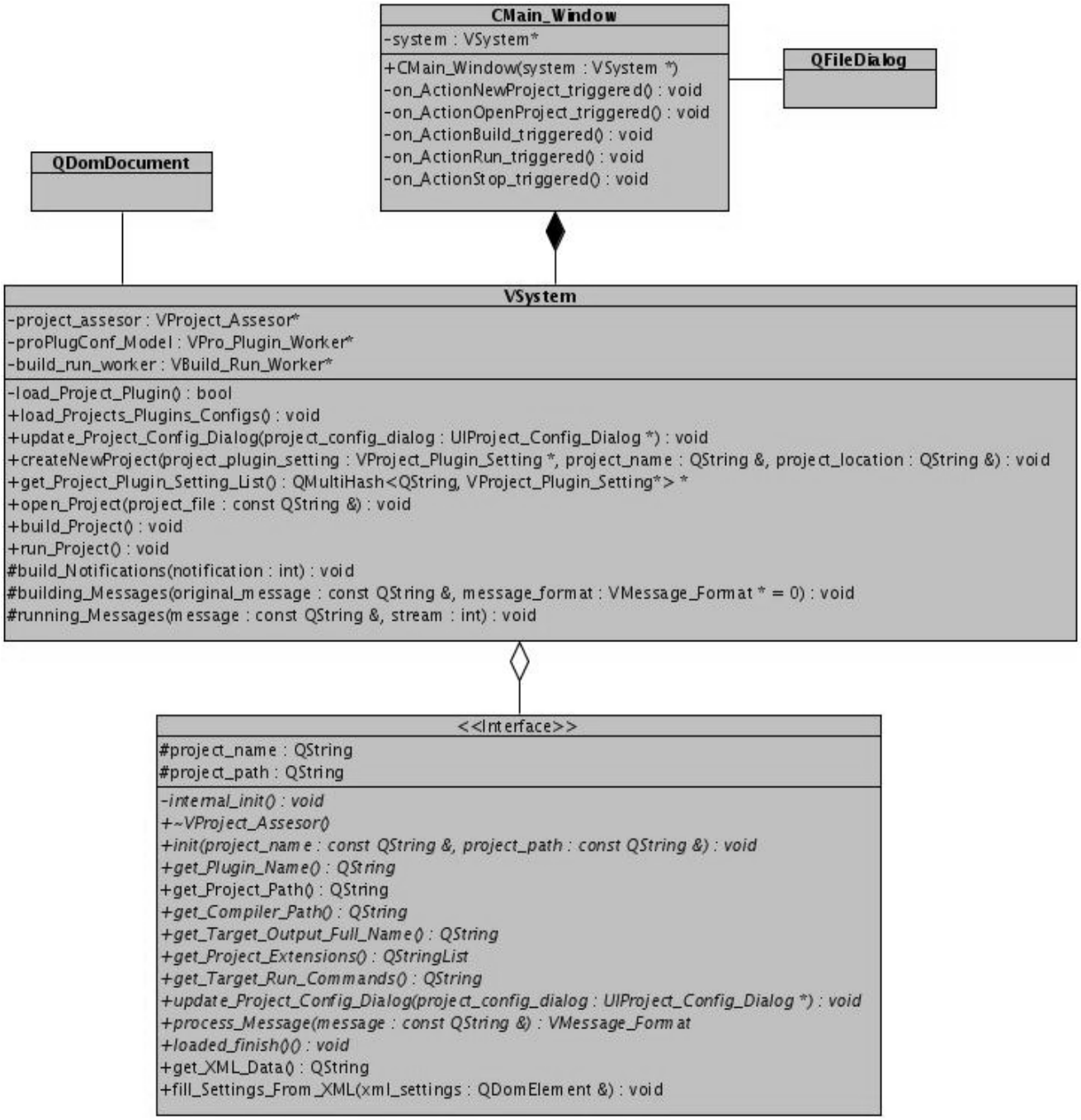



**Figura B1 Clases del Diseño CU Abrir Proyecto**

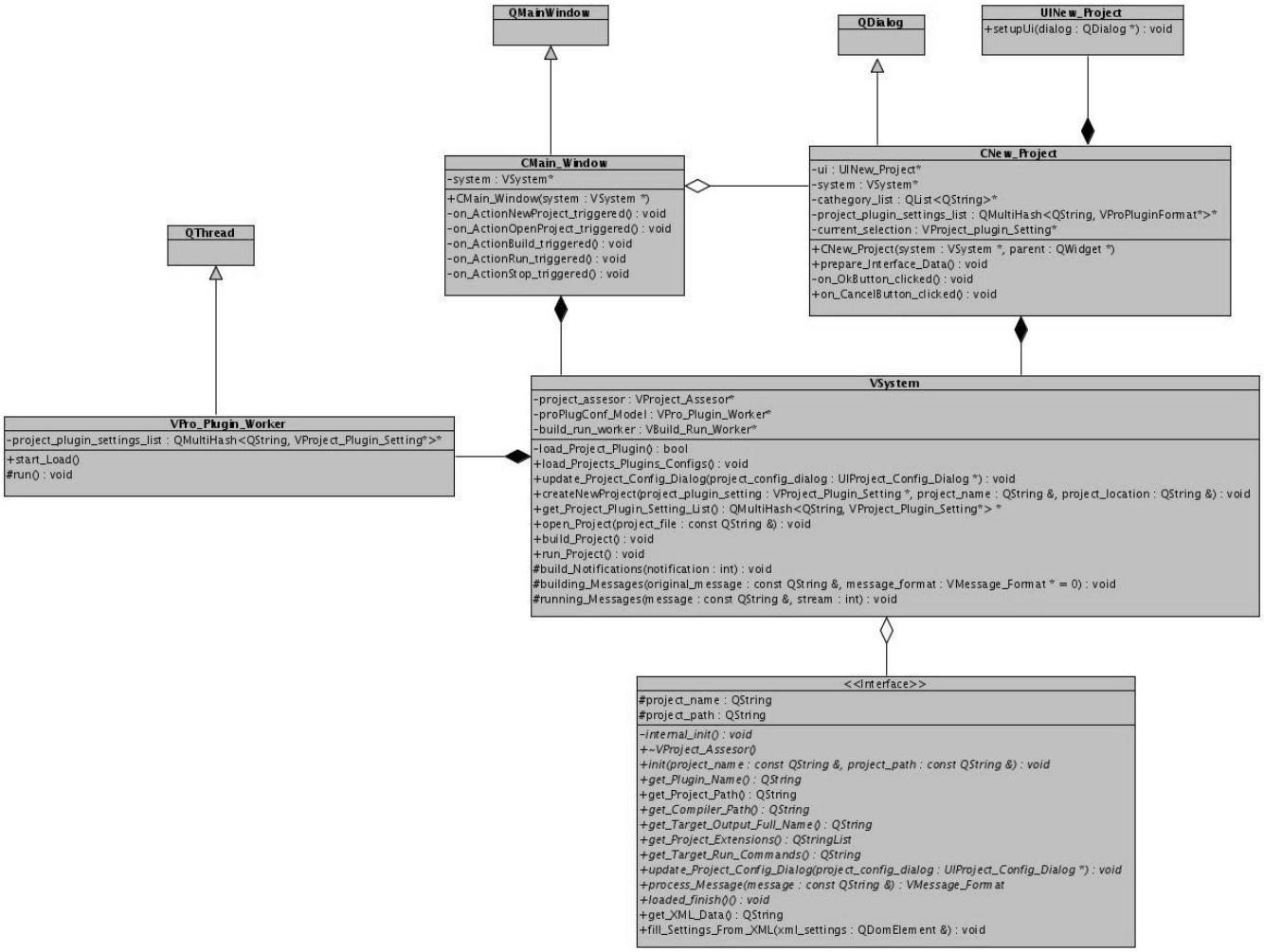



**Figura B2 Clases del Diseño CU Nuevo Proyecto**

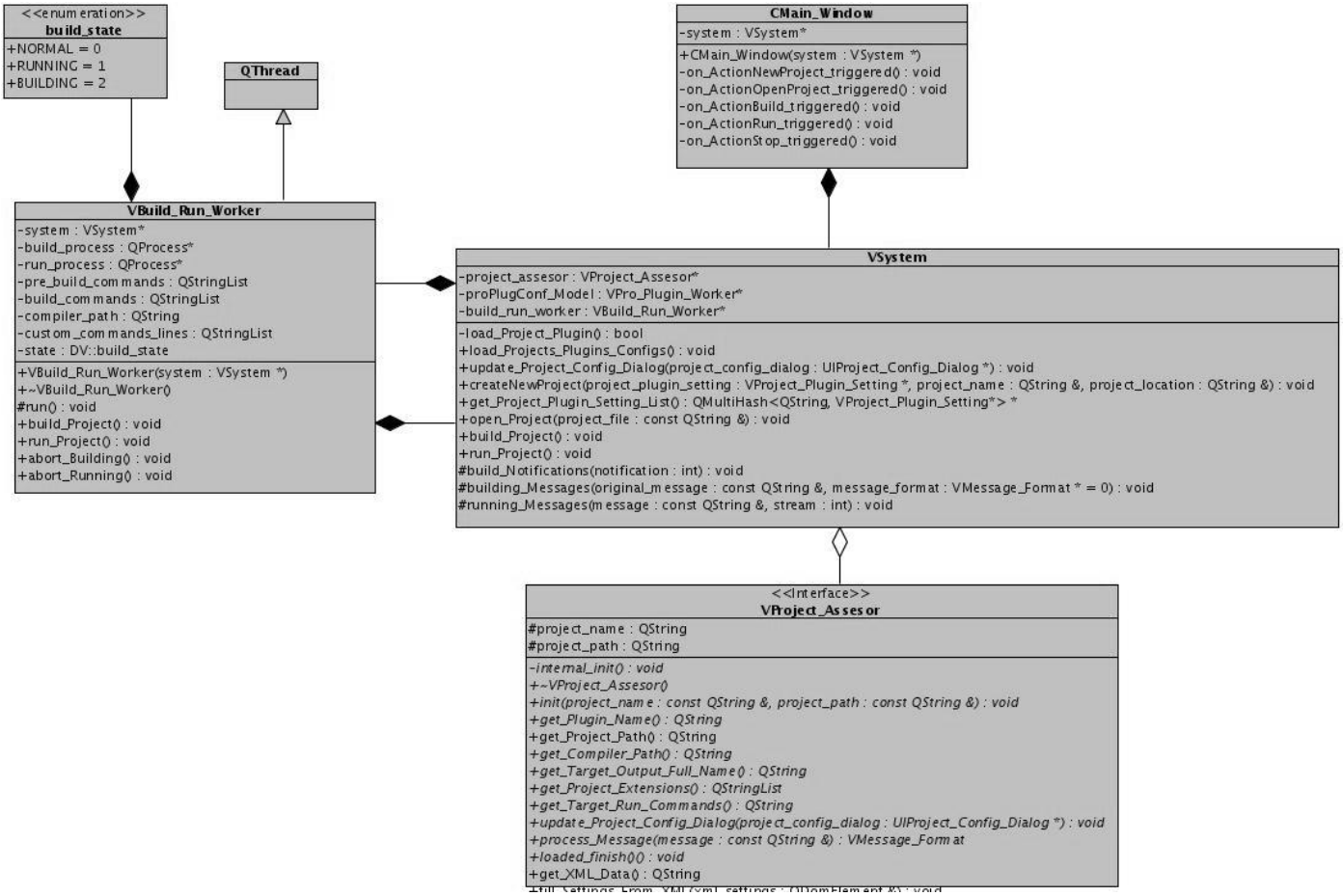



**Figura B3 Clases del Diseño CU Construir Proyecto**

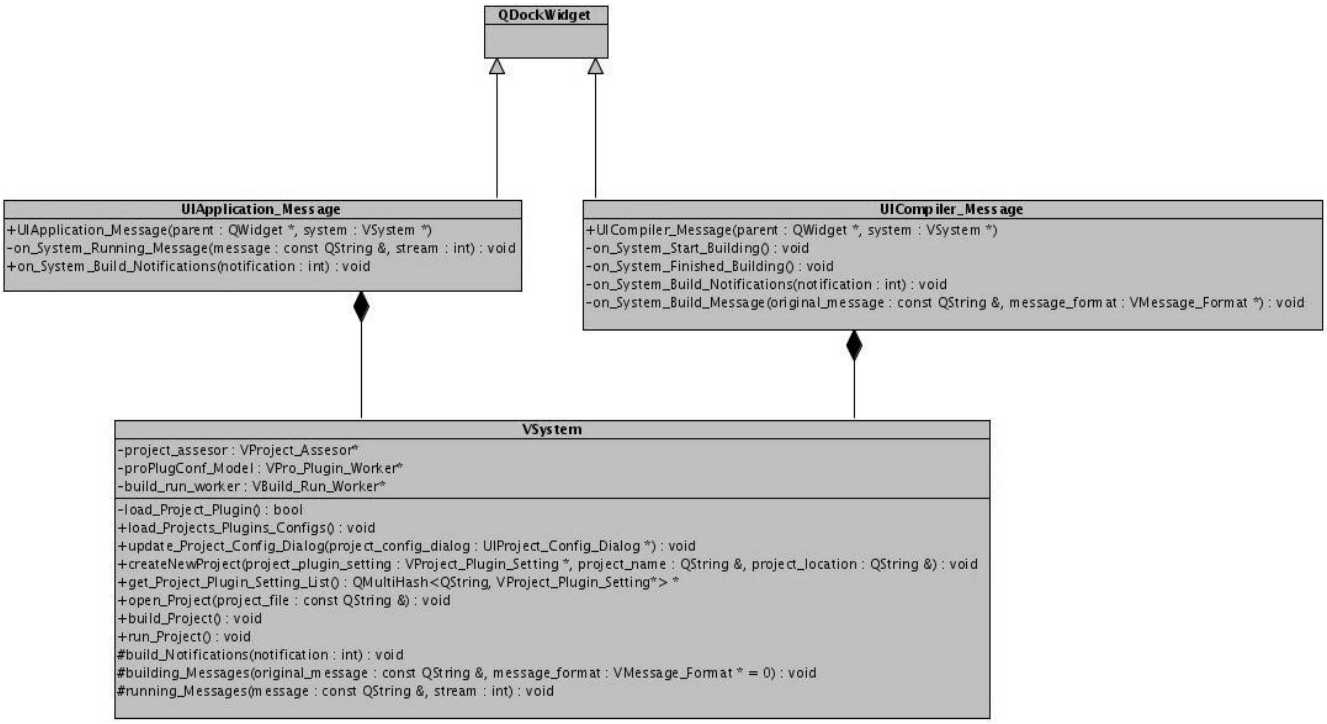



**Figura B4 Clases del Diseño CU Gestionar Mensajes**

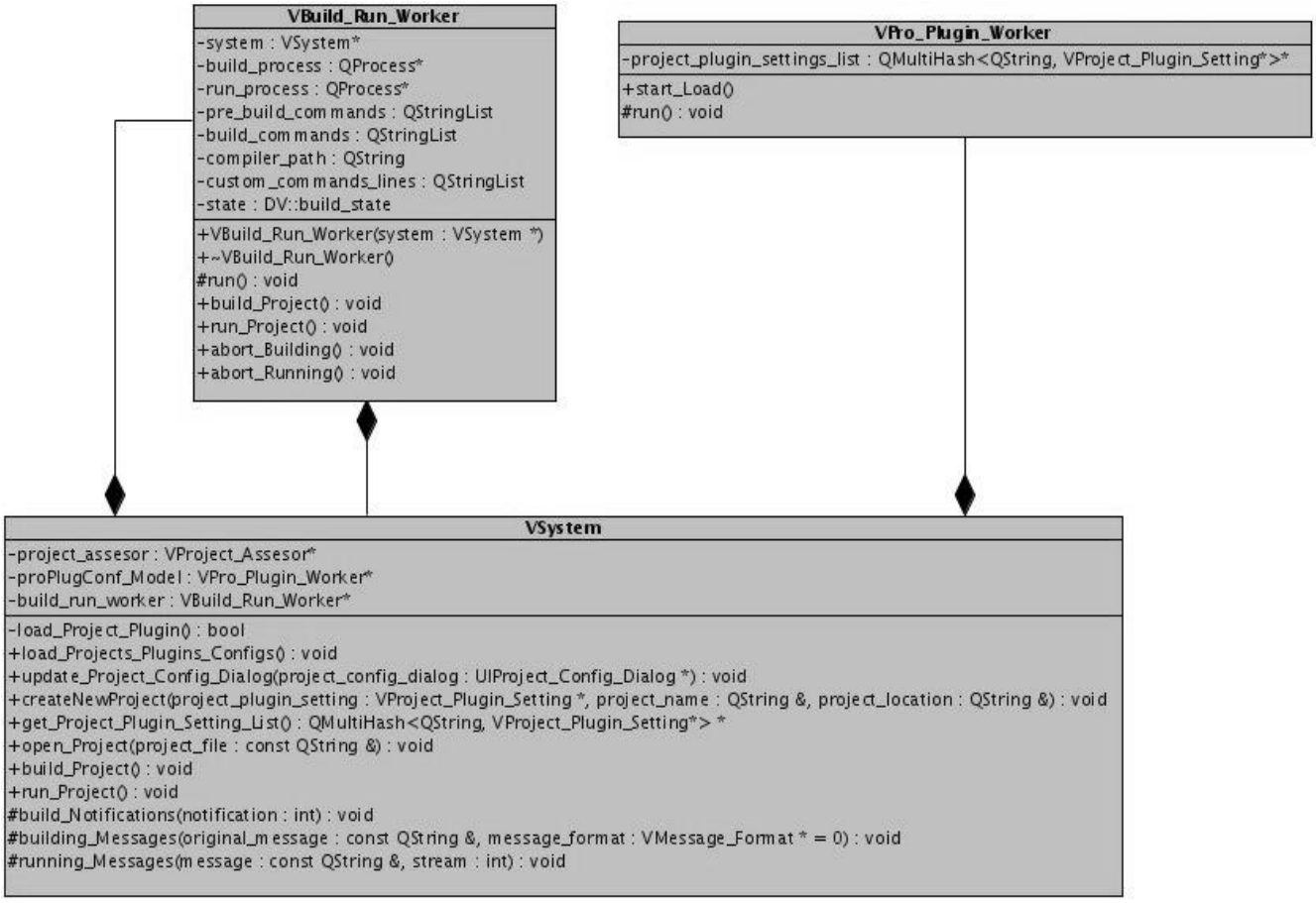



**Figura B5 Clases del Diseño Núcleo del Sistema**

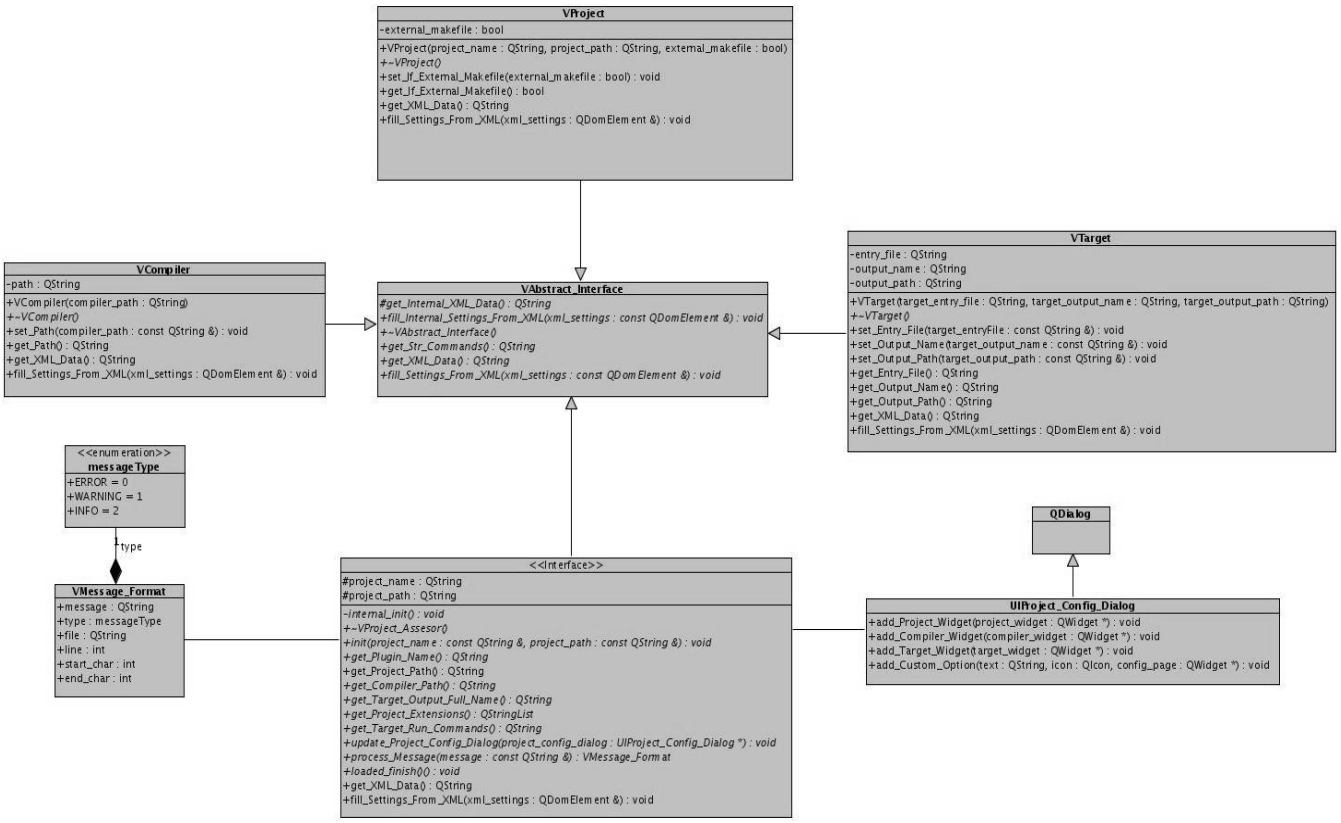



**Figura B6 Clases del Diseño Sistema de Plugins**

## ANEXO C. Estudio de Factibilidad

| Tipo de Actor | Descripción | Factor de Peso |
|---|---|---|
| Simple | Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de programación (API, Application Programming Interface ) | 1 |
| Medio | Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto | 2 |
| Complejo | Una persona que interactúa con el sistema mediante una interfaz gráfica | 3 |

**Tabla C 1 Cálculo de UAW**

| Tipo de Caso de Uso | Descripción | Factor de Peso |
|---|---|---|
| Simple | El Caso de Uso contiene de 1 a 3 transacciones | 5 |
| Medio | El Caso de Uso contiene de 4 a 7 transacciones | 10 |
| Complejo | El Caso de Uso contiene más de 8 transacciones | 15 |

**Tabla C 2 Cálculo de UUCW**

| **Factor** | **Descripción** | **Peso** | **Valor asignado** | **Comentario** | **Peso x Valor** |
|---|---|---|---|---|---|
| T1 | Sistema distribuido | 2 | 0 | El sistema es centralizado | 0 |
| T2 | Objetivos de rendimiento o tiempo de respuesta | 1 | 5 | El sistema debe responder con la mayor velocidad posible frente a las acciones del usuario. | 5 |
| T3 | Eficiencia del usuario final | 1 | 5 | El sistema debe hacer uso óptimo de la memoria del ordenador, efectuando procesos como la compilación de un proyecto de forma rápida. | 5 |
| T4 | Procesamiento | 1 | 1 | No se realizan cálculos complejos. | 1 |

| | | | | | |
|---|---|---|---|---|---|
| | interno complejo | | | | |
| T5 | El código debe ser reutilizable | 1 | 5 | Se requiere que el código sea lo más reutilizable posible. | 5 |
| T6 | Facilidad de instalación | 0.5 | 3 | El sistema posee poco requerimientos para la instalación. | 1.5 |
| T7 | Facilidad de uso | 0.5 | 5 | El sistema será fácil de utilizar. | 2.5 |
| T8 | Portabilidad | 2 | 5 | El sistema será portable a varios sistemas. Multiplataforma | 10 |
| T9 | Facilidad de cambio | 1 | 3 | Se requerirá un costo mínimo en mantenimiento y cambios. | 3 |
| T10 | Concurrencia | 1 | 4 | El sistema es concurrente. | 4 |
| T11 | Incluye objetivos específicos de seguridad | 1 | 0 | Es una aplicación de escritorio, para desarrollar multimedia, no utiliza herramientas de administración. | 0 |
| T12 | Provee acceso directo a tercera partes | 1 | 4 | Utiliza compiladores provistos por terceras partes. | 4 |
| T13 | Se requieren facilidades especiales de entrenamiento a usuarios | 1 | 3 | Sistema intuitivo, pero se requieren conocimientos de desarrollo de multimedia. | 3 |
| Total Σ( Peso x Valor Asignado ) | | | | | 49 |

**Tabla C 3 Cálculo de TCF**

| Factor | Descripción | Peso | Valor asignado | Comentario | Peso x Valor |
|---|---|---|---|---|---|
| E1 | Familiaridad | 1.5 | 4 | El equipo se encuentra bastante familiarizado con el modelo | 6 |
| E2 | Experiencia con la aplicación | 0.5 | 4 | El equipo lleva tiempo trabajando con este tipo de aplicación | 2 |
| E3 | Experiencia en orientación a objetos | 1 | 4 | El equipo programa orientado a objetos | 4 |
| E4 | Capacidad del analista líder | 0.5 | 3 | Tiene una experiencia media. | 1.5 |
| E5 | Motivación | 1 | 5 | El grupo está motivado | 5 |
| E6 | Estabilidad de los requerimientos | 2 | 3 | Se esperan algunos cambios | 6 |
| E7 | Personal part-time | -1 | 3 | El grupo trabaja a tiempo completo | -3 |
| E8 | Dificultad del lenguaje de programación | -1 | 3 | Se usará el lenguaje C++ | -3 |
| Total Σ( Peso x Valor Asignado ) | | | | | 18.5 |

**Tabla C 4 Cálculo de EF**