Universidad de las Ciencias Informáticas
Junio de 2009

Desarrollo de un IDE libre y multiplataforma para la creación de componentes visuales de ActionScript para Software Educativo: codeDraw

Autor: Jorge Antonio Díaz Gutiérrez

Tutores: Ing. Abel Ernesto Lorente Rodríguez
Ing. Roberto Ferrer Obregón

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas (UCI) a que haga el uso que estimen pertinente con el mismo.

Para que así conste firmo la presente a los ____ días del mes de Junio del 2009.

Autor:

Jorge Antonio Díaz Gutiérrez

_____

Tutores:

Ing. Abel Ernesto Lorente Rodríguez

_____

Ing. Roberto Ferrer Obregón

_____

“Si tú tienes una manzana y yo tengo una manzana y las intercambiamos, entonces ambos aún tendremos una manzana. Pero si tú tienes una idea y yo tengo una idea y las intercambiamos, entonces ambos tendremos dos ideas.”

George Bernard Shaw

Escritor Irlandés (1856-1950)

Dedicatoria y Agradecimientos

A mi mamá, que nunca ha dejado de confiar en mis decisiones y me ha apoyado toda la vida.

A mi papá, que no ha dejado nunca de forjar en mí los valores de un ser humano consecuente con sus tiempos.

A mi hermano, que nunca tuvo “un Roly” que lo guiara como él me guió a mí, como me dice él.

A los 3 que los quiero con la vida y les dedico el título que acompañará eternamente mi nombre, de ellos también es, aunque no sepan programar.

A mi abuela, a todos mis tíos, a mis primos, a los 2 sobrinitos nuevos que andan llorando por ahí, que hacen entre todos una familia espectacular y me la ponen difícil a la hora de organizarlos en esta dedicatoria. Todos valen mucho y no es justo que los ordene en un listado.

A mi abuelo Jorge, que lo extrañamos mucho.

A mis abuelos paternos que casi no pude conocer.

A mi novia Lissette, que me ha apoyado mucho durante estos 5 años en la universidad y que quisiera que fuera toda la vida, ella lo sabe.

A su familia, que ya forma parte de la mía.

A mis 4 amigos/hermanos/mariscos inseparables: Humberto, José, Fabián y Eric, que nos debería dar un título para los 5 juntos, cada uno tenemos mucho de los otros.

Al Migue, que es uno de esos amigos para toda la vida.

A la gente de mi grupo, que mucho hemos navegado para que este año estemos casi todos los que empezamos.

A mis profesores, que día a día lucharon conmigo por este momento. A todos ellos, desde primaria hasta ahora.

A mis tutores, Abel y Roberto, que me apoyaron mucho para hacer este trabajo aún cuando, desde un inicio, eran “detractores” del mismo.

Al tribunal, que se ha portado genial en cada corte: crítico y objetivo siempre.

A Dosagues que nos trazó el camino desde 1º año. A “Renier PC”, a la profe Lombillo y a todos los otros que han tenido que ver de alguna forma u otra.

A la gente del laboratorio: los “profes mariscos” que no dejaron nunca de ayudar y a los que aún les falta para graduarse y van por muy buen camino.

A Roberto, ahora porque que es un gran amigo, un gran profesional y no dudo que un gran compañero de trabajo cuando unamos Primavera con GALM en Septiembre de 2009.

A Yonnys, Michel, Geyser, al viejo Yosber y a Joan Pablo, por la ayuda desde que hace 3 años compartimos laboratorio.

A todos mis amigos del pre, que bastante lloramos el día en que nos tuvimos que separar.

A la gente anónima de los foros y de los comentarios en los Blogs, que ayudan a mejorar lo poco que es uno y comparten sus conocimientos con todos.

A la gente de Adobe por liberar el SDK de Flex en el 2008, sino, estuviera yo haciendo otra tesis.

A todo el que me faltó por mencionar más arriba y que me ayudó en esto. Son muchos los que han formado parte de estos 5 años.

A todos ellos, muchas gracias…

Resumen

En la Universidad de las Ciencias Informáticas (UCI) el proceso de desarrollo de software educativo involucra el uso de herramientas privativas para la producción de los mismos, lo que imposibilita la comercialización internacional de los productos. El presente trabajo consiste en el desarrollo de un ambiente de desarrollo libre y multiplataforma que persigue el propósito de reemplazar los que se utilizan actualmente y lograr introducir el software educativo cubano en este mercado. El documento recoge, en un primer momento, el estudio del estado del arte y la investigación realizada. Luego detalla el proceso de desarrollo de software seguido para la creación de la solución y los aspectos técnicos de la creación del mismo.

Índice

Introducción

Introducción

El siglo XXI es conocido por los expertos como “el inicio de la era de la información”. Es la etapa del desarrollo de la humanidad en la que la globalización de los conocimientos, las noticias y el dinamismo constante de tecnologías son un hecho. La Internet ha revolucionado la concepción tradicional de la comunicación, del aprendizaje, de las noticias, de la economía y del mundo en general.

La Universidad de las Ciencias Informáticas (UCI) juega un papel muy importante en este nuevo movimiento que, a nivel nacional, igualmente se está desarrollando. Como universidad creada bajo la concepción de fomentar el aprendizaje conjuntamente con la producción, es la guía nacional para la investigación de nuevas tecnologías, metodologías de trabajo y la creación de soluciones que estandaricen la producción en el país. La naciente industria de software cubana cuenta con muchos retos sociales y comerciales para su avance en la labor de informatizar a su pueblo y convertir al sector en uno de los principales rubros de ingresos de recursos al país.

Sucede que, al igual que la mayoría de las esferas de la sociedad cubana, la industria del software se ve afectada por un conjunto de leyes de la administración de exportación gobierno de los Estados Unidos de América (EEUU), conocidas internacionalmente como el “bloqueo económico a Cuba”. Las herramientas que se vienen utilizando desde hace años para la producción de diversas soluciones informáticas, las de mayor calidad en el mercado internacional, no son vendidas de manera legal a las instituciones cubanas y las alternativas libres, que realmente resulten factibles para crear esos tipos de software, generalmente escasean. Son muy pocos los países que sufren el injusto bloqueo económico por parte de la potencia más grande que se ha conocido. Todos, países del tercer mundo con pocas capacidades tecnológicas para crear soluciones alternativas a situaciones como esta.

Aún así, Cuba es un país que, por muchas limitaciones comerciales que agredan a la economía, ha decidido no quedarse atrás en este tema, por lo que se ha trazado planes de migración hacia nuevas plataformas que garanticen la independencia tecnológica y la libertad de comercio de software con el mundo.

El software educativo, como concepto surgido en las últimas décadas, juega un papel muy importante en esta revolución. En el país se ha comenzado a desarrollar desde hace varios años, con el objetivo de educar más aún al pueblo, partiendo desde los propios centros escolares y demostrando muy buenos resultados. Esta política es totalmente contraria a la que muchas empresas del mundo llevan, donde la enseñanza es una cuestión comercial y la venta de

programas para la educación cuenta con precios exageradamente altos que a su vez atan a los clientes a la compra de futuras versiones, con escasos contenidos adicionales que no incluyen para tener “nuevas mejoras”.

En la sociedad cubana, el software educativo se adopta a través de planes gubernamentales dirigidos por los ministerios pertinentes con el objetivo de apoyar los métodos tradicionales de enseñanza en todos los niveles educacionales. Es por ello que, la calidad del mismo, es uno de los aspectos más importantes a tener en cuenta en el momento de crearlos. Los términos interactividad y multimedia, actualmente son inherentes a lo que se le podría llamar “un buen software educativo”. La capacidad de respuesta de la aplicación a las acciones de los estudiantes o usuarios y la complementación del mismo con imágenes, audios y videos, son, tan medidores de la calidad del mismo, como lo es el contenido pedagógico que respalda la organización de estos en el producto.

La interactividad de un software viene respaldada por un ambiente en el cual el usuario se siente familiarizado con el tema del producto. Los componentes visuales que acompañan su navegación a través de los diferentes tópicos de la aplicación, juegan el papel más importante de los métodos didácticos de aprendizaje digital, ya sean juegos, ejercicios, mensajes, avisos e instrucciones. Son los que le dan la vida gráfica al entorno del software y contienen “los lugares” en donde los estudiantes encuentran la información.

Lamentablemente, las herramientas profesionales de mayor uso internacional para la creación de estos componentes visuales, las cuales se utilizan en la UCI, están incluidas en las prohibiciones del bloqueo de los EEUU. La creación de proyectos en la universidad que exploren la creación de alternativas libres es una necesidad y, a su vez, el uso de herramientas libres que permitan estandarizar el proceso sobre una vía totalmente legal. También lo es aprovechar tecnologías existentes para crear este tipo de software y no caer en ciclos de desarrollo de nuevas plataformas que durarían años de creación y desaprovecharían la posibilidad de proveer a la industria cubana de software de soluciones prácticas a corto plazo. A todo esto se suma que en la UCI abundan las prácticas de programación en ActionScript para la creación de software educativo, otro de los aspectos claves a tener en cuenta para la creación de una nueva herramienta.

La **situación problémica** gira en torno a que, actualmente, la Universidad de las Ciencias Informáticas (UCI) desarrolla software educativo (en adelante SWE) para instituciones nacionales utilizando Adobe Flash y Macromedia Director, herramientas privativas con licencias que no son reconocidas por clientes potenciales en el extranjero. El bloqueo impuesto por el gobierno de los Estados Unidos a Cuba incluye la prohibición de venta de licencias comerciales de estas herramientas de desarrollo, lo que limita totalmente su empleo en productos con destino a la exportación. A esto se suma la falta de entornos de desarrollo multiplataforma para la creación de este tipo de aplicaciones, lo cual frena la estrategia nacional de migración hacia la independencia tecnológica y el software libre.

Todo lo mencionado anteriormente trae su consecuente pérdida de ganancias y la necesidad de uso de intermediarios o terceros que ubiquen estos productos en el mercado internacional, a lo que se suma las implicaciones económicas que todo esto acarrea.

La principal deficiencia se encuentra en la necesidad que existe de una herramienta libre capaz de ofrecer un ambiente de desarrollo para la creación de componentes visuales de software educativo y que no desaproveche las prácticas existentes de programación en ActionScript que abundan en la universidad.

Es por ello que se evidencia el siguiente **planteamiento del problema científico:**

¿Cómo desarrollar componentes visuales de ActionScript para Software Educativo con herramientas libres?

El **campo de acción** de la investigación va dirigido hacia el desarrollo de software educativo y productos multimedia en la UCI y el **objeto de estudio** de la misma es el desarrollo de un IDE para la creación de componentes visuales de SWE en formato SWF utilizando ActionScript.

Con el fin de dar solución a la problemática antes mencionada se define como **objetivo general** desarrollar un IDE que permita crear componentes visuales de ActionScript para software educativo, del cual se derivan los siguientes **objetivos específicos**:

1. Crear el framework gráfico para el núcleo del entorno visual.
2. Elaborar el entorno de desarrollo visual.
3. Optimizar la herramienta, tomando en consideración los criterios de otros desarrolladores cada vez que se termine una iteración.
4. Elaborar un tutorial para facilitar la familiarización del usuario con la nueva

herramienta.

5. Documentar el trabajo realizado.

Para cumplir los objetivos específicos, se plantearon las siguientes 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.Investigar la estructura de los núcleos gráficos de ActionScript 3.0

6. Análisis de las diferentes herramientas existentes de creación de componentes visuales para SWE, fundamentalmente en el mundo del software libre.

7. Estudio de los posibles componentes a utilizar que permitan un conjunto de funcionalidades para la gestión de código.

8. Evaluar soluciones y herramientas encontradas.

9. Selección y fundamentación de una metodología de desarrollo.

10. Levantamiento de Requisitos.

11. Revisión de Requisitos.

12. Definir Casos de Uso./Historias de Usuario

13. Detallar Casos de Uso/Historias de Usuario

14. Estructurar Modelo Casos de Uso (en dependencia de la metodología a utilizar)

15. Realizar un prototipo de interfaz de usuario.

16. Evaluación de los patrones de arquitectura.

17. Seleccionar un IDE de desarrollo.

18. Implementación de un prototipo no funcional.

19. Realizar el análisis y diseño.

20. Implementación de los Casos de Uso más importantes.

21. Documentar todo el trabajo realizado.

Y finalmente, para guiar el desarrollo de la solución se plantea la **idea a defender**: si se crea un entorno de desarrollo visual que permita diseñar gráficos y generar el código ActionScript 3.0 que los representa, se podrán desarrollar componentes visuales de SWE fácilmente sin necesidad de utilizar herramientas privativas, lo cual posibilitará un incremento de la productividad y el ahorro considerable del tiempo de trabajo.

Capítulo 1

Herramientas para la creación de componentes visuales de ActionScript

En el presente capítulo se realizará la fundamentación teórica: Primeramente se expondrá el estado del arte de las soluciones existentes y se analizarán las causas de éxito y de fracaso de estos proyectos. Luego se introducirá el tema de las tendencias tecnológicas de la solución a proponer y se explicarán detalladamente los aspectos a tener en cuenta en la selección de las herramientas, tecnologías y metodología a utilizar.

1.1 Conceptos generales relacionados

Componentes visuales: Todo objeto gráfico que esté orientado a almacenar respuestas a acciones de entrada sobre la interfaz de usuario. Igualmente pueden ser de interacción o solamente de lectura de información. Ejemplos comunes son los botones, los cuadros de texto, formularios de inscripción, botones de ejercicios, dibujos, etc.

IDE: Un Ambiente o Entorno de Desarrollo Integrado (del inglés “Integrated Development Enviroment”) es una herramienta que integra un conjunto de aplicaciones para crear otros softwares o otro tipo de contenidos digitales, ya sean imágenes, audios, videos, etc. En el desarrollo de software su principal objetivo es automatizar tareas, empaquetar el código, compilarlo, ejecutarlo y generalmente depurarlo, por mencionar algunos ejemplos. Por otro lado, en los IDEs para la creación de contenidos visuales se ofrece una serie de herramientas de dibujo, trazado, edición de propiedades, edición de color, etc.

Kit Estándar de Desarrollo: Conjunto de herramientas, generalmente compiladores y librerías de código, que permiten a un equipo de desarrollo, construir, de una manera rápida, software para una plataforma específica. No incorporan un IDE, sencillamente son una interfaz de programación a la cual se le pueden crear entornos de desarrollo que la aprovechen para agilizar más aún la producción.

Plataforma Flash: Nombre oficial que recibe el conjunto de herramientas, aplicaciones y librerías de componentes de Adobe y se encuentra centrada en Adobe Flash Player como reproductor final de los archivos SWF. De las herramientas, se encuentran en un primer plano el IDE de Adobe Flash y Adobe Flex Builder. Otras como Adobe Photoshop y Adobe Illustrator se encuentran vinculadas hasta cierto punto pero no juegan un papel fundamental. El lenguaje de programación que utiliza es ActionScript.

Software privativo: Se refiere a todo aquel software o fragmento de software cuya licencia comercial, de uso o de publicación no permite compartir el código fuente del mismo o, lo hace, pero restringe las libertades de uso del mismo.

Software libre: Hace referencia a la libertad de los usuarios o desarrolladores de tener acceso libre al código fuente para copiarlo, leerlo, distribuirlo, cambiarlo y mejorarlo a su gusto. Existen diferentes variantes. La principal, promovida por la Free Software Foundation, es la más utilizada en el mundo, respaldada legalmente por la licencia GNU/GPL (del inglés "General Public License"). Existen otras variantes de código abierto como Mozilla Public License, L-GPL, etc.

1.2 Análisis de las soluciones existentes

La creación de componentes visuales es uno de los aspectos más importantes para el desarrollo de un software educativo multimedia. Lamentablemente, las herramientas utilizadas en la actualidad, se encuentran centradas hacia plataformas Microsoft Windows, además de que no venden licencias legales a Cuba. Aún, en caso de existir la posibilidad de compra, los precios son muy elevados.

Las alternativas libres y gratuitas no escasean, aunque, entre estas, las que resultan factibles sí. La desunión que existe a lo largo del orbe en el desarrollo de proyectos software libre de pequeña envergadura, finalmente deja con un pequeño conjunto de soluciones, las cuales se describen a continuación:

1.2.1.AJAX Animator: Un IDE on-line creado solamente para animaciones y diseño. Es capaz de exportar el resultado final a Adobe Flash Player (en adelante Flash Player), Microsoft Silverlight y archivos GIF. Al ser desarrollado en una plataforma WEB, no limita al usuario a un sistema operativo específico y depende solamente de las capacidades del navegador del desarrollador. No ofrece control de los componentes gráficos una vez que son generados y lamentablemente el creador planteó que el proyecto está descontinuado. No presenta integración a ningún lenguaje de programación. (AJAX Animator Dev, 2008)

1.2.2.Salasaga Creado por Justin Cliff en 2007, es una alternativa para la creación de presentaciones demostrativas que ofrecen, utilizando capturas de pantallas, comentarios y sencillas figuras que complementan los tutoriales. El objetivo es similar al que cumplen Adobe Captivate y Tech Smith Camtasia Studio, aunque de una manera mucho más arcaica. El resultado final es exportado a un archivo de Flash Player y la post-edición de los contenidos tampoco es posible. Aún así, Salasaga cuenta con una buena aceptación y es una alternativa aceptable para crear este tipo de presentaciones educativas. Está desarrollada sobre C y utiliza las librerías gráficas de GTK, por lo que puede ser utilizado en los sistemas operativos más populares sin problema alguno. No posee integración con el lenguaje ActionScript. (Cliff, 2008)

1.2.3.OpenDialect: Un IDE para crear presentaciones principalmente y Aplicaciones Enriquecidas de Internet (del inglés “Rich Internet Application”, en adelante RIA) creado con Adobe Flex y C#. Siendo muy fácil de portar para Linux, a través del uso de las librerías Mono, OpenDialect cuenta con 17 componentes visuales prediseñados que permite al desarrollador crear presentaciones y organizar la aparición de las mismas desde un control de línea de tiempo. Exporta el trabajo final a archivos Flash Player y, en su última versión, a Adobe AIR. Cuenta con un sencillo editor de ActionScript 3.0 de pocas prestaciones para el trabajo con el código. Su principal deficiencia se encuentra en que la librería de componentes se limita solamente al uso de los que vienen incluidos en la aplicación y adicionar otros, gráficamente, es imposible. (Open Dialect Dev, 2008)

1.2.4.Flash For Linux (en adelante f4L): Fue una de las primeras herramientas que surgieron como alternativas a Macromedia Flash MX. Nació de las intenciones de un estudiante universitario turco de utilizarla en su centro. Como herramienta de animación, tuvo mucho éxito inicialmente, pero la falta de divulgación llevó al abandonado del proyecto por parte de sus desarrolladores en el año 2005. La última versión permite exportar archivos SWF versión 5 y no se integra a ningún lenguaje de programación. (f4l Team, 2005)

1.2.5.Qflash: El principal objetivo de Qflash fue crear una copia exacta de Macromedia Flash MX para Linux. Se desarrolló en C++ con el uso de las librerías gráficas Qt, de ahí su nombre. El proyecto no tuvo éxito y fue abandonado para, conjuntamente con los desarrolladores de f4L, crear el proyecto UIRA. (Sergi Pérez Maña, 2007)

1.2.6.UIRA: Es el resultado de la unión de los desarrolladores de f4L con los de Qflash en el año 2006. En sus inicios tendía a ser más ambicioso que el propio Macromedia Flash, en cuanto a capacidades técnicas se refiere, lo que llevó a un trágico fin cuando, casualmente en enero de 2008, Francia aprobó la nueva ley sobre derechos de autor, entrando el proyecto en problemas legales y dejando, solamente, el núcleo de una interfaz de usuario con limitadas funcionalidades. Actualmente, los desarrolladores iniciales, en la página oficial del proyecto, convocan a nuevos integrantes a continuar el desarrollo del producto y lograr una primera versión. (Florian Delizy, 2007)

1.2.7.Ktoon: De procedencia latinoamericana, específicamente de Colombia, Ktoon es una herramienta para crear animaciones en 2D con un enfoque profesional. El objetivo principal de sus creadores, Toonka Films, era fomentar el desarrollo de la industria de la animación y el diseño en su país. Posee una buena gestión de los gráficos, la línea de tiempo y las capas de dibujo, lo que lo hace una de las mejores herramientas libres para crear contenido gráfico para Flash Player,

pero carece de interactividad y de control de los componentes una vez generado el producto final. No se integra a ActionScript. (Toonka Films, 2008)

1.2.8. Picassus: Creado en la Universidad de las Ciencias Informáticas de Cuba, Picassus es el primer producto público del proyecto GALM (siglas de "*Grupo de Alternativas Libres para Multimedia*"), el cual consiste en un ambiente de desarrollo para crear archivos SWF con animaciones fotograma a fotograma (Martín & Saez, 2009). El proyecto se encuentra en desarrollo y promete un buen futuro, pero su principal debilidad se encuentra en que está desarrollado sobre las librerías "Ming", por lo que incluir código es muy difícil, debido a que estas no soportan grandes volúmenes de ActionScript. (Ming, 2008)

La mayoría de estas alternativas no poseen la capacidad de integrarse con un lenguaje de programación que permita desarrollar la interactividad del producto, como es el caso de ActionScript. Son intentos que se limitan exclusivamente a la parte visual del IDE y se encuentran con barreras que hacen imposible la integración al código. Casi todos los proyectos fueron abandonados y, los pocos que continúan en pie, eliminaron el paso de integración de sus objetivos.

Adicionalmente, la aparición de ActionScript 3.0 hace poco más de 2 años, como es de entender, hace que la mayoría de estas herramientas no trabajen con el mismo, por lo que es necesario analizar, de forma separada, las alternativas existentes. Estas se desarrollaron de forma paralela a los entornos de desarrollo anteriores, pero hicieron mayor énfasis en el trabajo con el lenguaje de programación:

1.2.9.MTASC: Motion Tween ActionScript Compiler es el primer compilador de ActionScript libre. Creado por Nicolas Cannasse en el año 2005, MTASC es popularmente conocido por su velocidad de compilación, la cual se calcula hasta 20 veces superior al compilador de Adobe Flash. Es una aplicación de tipo consola, muy utilizada por editores de código libres de lenguaje ActionScript como FlashDevelop y SEPY ActionScript Editor. Es muy estricto en el tratamiento del lenguaje, característica que, indirectamente, ha contribuido al fortalecimiento de las aplicaciones para Flash Player. Su principal desventaja se encuentra en que no compila código ActionScript 3.0, solamente 2.0. A pesar de la estabilidad de la versión que se distribuye con los editores de código mencionados anteriormente y que también puede ser descargada desde su sitio oficial, el proyecto ha sido descontinuado debido a que su desarrollador ha creado un compilador totalmente nuevo llamado haXe, el cual se describe a continuación. (Motion Tween, 2007)

1.2.10.haXe: haXe es un compilador muy novedoso que incorpora el concepto de generar los lenguajes de cliente y servidor de manera simultánea para una aplicación WEB utilizando el

lenguaje de programación haXe. El mismo se basa en un concepto reciente, el cual propone su propio creador, en donde el código, luego de ser interpretado, genera el código ActionScript, JavaScript y PHP de la aplicación. Al igual que MTASC, no se ejecuta sobre una interfaz visual y se integra con una máquina virtual llamada Neko, para la extensión de funcionalidades. La versión 3.0 de FlashDevelop lo incorpora en su lista de compiladores. (Motion Tween, 2008)

1.2.11.SWFMill: Al igual que MTASC y haXe, SWFMill es un compilador a modo consola para crear archivos SWF y proveer con librerías a otros proyectos. El proceso de compilación consiste en leer un archivo en lenguaje XML que contiene la información sobre la ubicación de los contenidos y las propiedades de los mismos al ser insertados en el área visual. Es una buena alternativa para la creación de componentes visuales, pero no soporta trabajar con ActionScript 3.0. (Daniel Fischer, 2008)

1.2.12.Adobe Flex OpenSource SDK: El Kit Estándar de Desarrollo (en adelante SDK) de Adobe Flex (en adelante Flex), está compuesto por el compilador de ActionScript 3.0, las librerías de Flex y el depurador, los cuales son el núcleo para creación de contenido Flash Player, el eje principal de la plataforma que creada por Adobe Systems Inc.(en adelante Adobe). Es la más reciente y robusta de las alternativas libres, ya que es una cuestión institucional el desarrollo y la promoción del uso del SDK. Por otro lado, Adobe es la empresa creadora de Flash Player y del lenguaje ActionScript 3.0, lo que hace a su compilador el más fiel a cuestiones claves como estabilidad, flexibilidad y capacidad de evolución a futuras versiones de la tecnología. Su funcionamiento es a modo consola y es una herramienta totalmente multiplataforma. Se encuentra integrado a FlashDevelop y a Adobe Flex Builder, el IDE oficial de Adobe para trabajar con el SDK y crear RIAs. (Adobe Systems Inc., 2008b)

Al realizar el análisis de estos compiladores, se observa que MTASC y SWFMill no son capaces de trabajar con la versión más reciente de ActionScript. Por otro lado, haXe y Flex sí, aunque el primero cuenta con poco tiempo de creado y su objetivo principal dista mucho de la concepción de la herramienta de Adobe, ya que utiliza un lenguaje de programación propio para generar código ActionScript 3.0.

1.3 Tendencias y tecnologías

Históricamente, los IDEs del tipo “lo que ves es lo que obtienes” (del Inglés “what you see is what you get”, en adelante WYSIWYG) son los que mayor impacto y mejores resultados han demostrado tener en el desarrollo de componentes visuales.

WYSIWYG se logra a partir de la emulación gráfica, en tiempo de desarrollo, del código fuente antes de ser compilado o interpretado. De esta forma es posible crear y editar componentes en un entorno visual fácilmente y con las características gráficas deseadas desde un inicio, sin tener que recurrir a la supuesta satisfacción que generalmente resulta de hacer el mismo trabajo durante horas solamente utilizando código. La mayoría de las herramientas de este tipo son capaces de ser utilizadas por usuarios comunes sin el más mínimo conocimiento de programación.

En el análisis realizado anteriormente, las soluciones existentes no ofrecen incorporación con ActionScript 3.0 en tiempo de desarrollo. Las capacidades de edición visual de los mismos son logradas utilizando lenguajes como C y C++ conjuntamente con sus respectivas librerías gráficas.

A diferencia de esto, Adobe Flash, herramienta líder entre los entornos de desarrollo para Flash Player y la principal guía a seguir por los proyectos de software libre, ofrece, en tiempo de desarrollo, una interfaz muy amigable en el interior del ambiente de trabajo para lograr WYSIWYG. En Adobe Flash los componentes visuales son editados de manera gráfica y el código que los representa, paradójicamente, se mantiene oculto a los desarrolladores. A esto se suma que, si se utiliza código solamente, con un objetivo similar, las librerías de ActionScript 3.0 ofrecen una interfaz de dibujo que no permite controlar, luego de creados, los detalles gráficos para la edición de los mismos, lo que obliga a eliminarlos visualmente y volverlos a trazar con los datos nuevos.

1.4 Tecnología a utilizar

Según lo analizado anteriormente, el principal factor que propició el estancamiento del desarrollo de las soluciones existentes de software libre, es el momento de integración del entorno de desarrollo visual con los compiladores de código ActionScript. La incompatibilidad de los IDEs WYSIWYG con el lenguaje de programación interno de Flash Player solo logró resultados que se limitan a generar archivos SWF con animaciones que carecen de interactividad, una de las exigencias fundamentales de un software educativo.

Los equipos de desarrollo de las alternativas descritas anteriormente, tomaron la decisión de no utilizar Flash Player debido a la poca capacidad que poseían las versiones anteriores del reproductor para interactuar con el sistema operativo del cliente. Esta es una de las cuestiones que llevó a algunos de los proyectos mencionados a crear entornos de desarrollo on-line y así lograr, al menos, una gestión controlada de la información persistente. Adobe Flash Player está creado para acceder, bajo ciertas restricciones, a la red e igualmente, de forma muy limitada, al sistema de archivos local, razón por la cual también resultaba muy contradictorio proponerse utilizarlo para crear una alternativa libre. Aún así, es erróneo seleccionar una tecnología distante

del mismo. Esto obliga a crear complejos puentes de comunicación entre una herramienta y otra, la causa principal de los fracasos que se mencionaron anteriormente. También lo es crear una plataforma totalmente diferente que desaproveche los recientes cambios legales que ha sufrido la misma y las novedosas soluciones que han aparecido, las cuales se describen a continuación:

El 25 de Febrero de 2008, caracterizado por algunos como una medida de "supervivencia tecnológica preventiva", Adobe Systems Inc.(Adobe) liberó, bajo la licencia Mozilla Public License(en adelante MPL), el núcleo del SDK de Flex. Se ha especulado que la decisión fue tomada debido al amplio avance del uso de AJAX, práctica no estándar, muy utilizada para la creación de RIAs con software libre y que presenta una gran competencia a la plataforma Flash de Adobe en la red. El SDK libre de Flex está compuesto por el compilador de ActionScript 3.0, el depurador y la librería de componentes visuales.

Conjuntamente, ese mismo día se hizo pública la primera versión estable de Adobe Integrated Runtime (en adelante AIR), una herramienta capaz de portar al escritorio cualquier aplicación Flash Player y proveerla de las funcionalidades ausentes que tiene para comunicarse con el sistema operativo cliente y para acceder a los servicios de redes. Desde entonces, la integración de Flash Player y AIR ha dado lugar a una enorme cantidad de aplicaciones que van desde sencillos lectores de canales de noticias hasta clientes de televisión on-line y herramientas de modelado de sistemas, por mencionar algunos ejemplos. (Adobe System Incorporated 2008a)

Actualmente, la integración de Flash Player, Flex y AIR se presenta como la mejor alternativa para el desarrollo de la solución en cuestión. El uso combinado de estos cuenta con un respaldo de documentación muy organizado y que Adobe ha utilizado para llevar adelante su plataforma. Las publicaciones digitales sobre las capacidades, la flexibilidad y la escalabilidad de los proyectos basados en Flex y AIR son enormes. A continuación aparecen las características adicionales de cada herramienta o tecnología que se tuvieron en cuenta para tomar tal decisión:

1.5 Adobe Flash Player

Adobe Flash Player es conocido como la tecnología por excelencia para crear contenidos interactivos para la WEB y el escritorio. Sus creadores han logrado construir una poderosísima plataforma sobre sus hombros, haciéndola capaz de enfrentar una amplia variedad de proyectos. A continuación aparecen las principales características que se tuvieron en cuenta para seleccionarlo para este:

- *Fidelidad de Flash Player como emulador WYSIWYG*: Ninguna aplicación es capaz de mostrar mejores pre visualizaciones de sus resultados que ella misma. La interfaz de programación de Flash Player (del inglés "Application Programming Interface", en adelante API) para dibujar, es

capaz de crear gráficos tan complejos como los que se producen desde el propio IDE de Adobe Flash.

- *Potencia del lenguaje de programación de las aplicaciones Flash Player*: ActionScript 3.0 ha sido el resultado de un análisis muy profundo de sus versiones anteriores y de la unión de muchos esfuerzos por crear un lenguaje altamente robusto para controlar interactividad, acceso a servicios de comunicación y una gestión organizada de contenido multimedia.

- *Respaldo aceptable de proyectos de software libre a nivel mundial*: Existen una buena variedad de frameworks y librerías de ActionScript 3.0 que ayudan a extender las soluciones finales de proyectos a lo largo del mundo. Uno de sus principales promotores es el propio Adobe.

Como desventaja principal se encuentra:

- *Adobe Flash Player no es un software libre*: Aún cuando el formato SWF es público, Adobe limita a los desarrolladores a solo utilizar Flash Player para ejecutarlos, prohibiendo la creación de reproductores similares. Sin embargo, la capacidad de extensión de la plataforma está muy bien concebida y la gama de herramientas es bastante amplia, aunque todo gire en torno al reproductor. (Adobe Systems Inc. 2008b)

1.6 Frameworks seleccionados para el desarrollo

1.6.1. Adobe Flex Open Source SDK

Flex es una tecnología reciente y de gran aceptación en el desarrollo de RIAs y aplicaciones para Flash Player. En poco menos de 3 años, ha inundado la red con soluciones que oscilan entre sencillas aplicaciones y grandes sistemas empresariales, gracias a la calidad de su arquitectura en general. Su gran impacto vino impulsado por la aparición de la versión 2.0. El framework actual está compuesto por una librería de componentes de interfaz de usuario que ofrecen un alto grado de interactividad y la capacidad de comunicarse con los sistemas de datos más comunes que existen.

Al igual que Flash Player, el SDK libre de Flex es desarrollado por Adobe y mantenido por diversas comunidades a lo largo del mundo que crean continuamente nuevos proyectos y mejoran los existentes. La principal línea de desarrollo de estas es la elaboración de nuevos componentes para las comunicaciones y de frameworks arquitectónicos para el desarrollo de aplicaciones on-line.

Su principal debilidad residía en que, al ejecutarse sobre Flash Player, se encontraba obstaculizado por las limitantes de acceso que presenta el reproductor, la cual fue eliminada con la aparición de AIR a inicios de 2008 haciendo posible que se pueda seleccionar a Flex como uno de los frameworks a utilizar. (Adobe Systems Inc. 2008c)

Flex es clave para la creación de la presente solución debido a que aporta la librería de componentes de interfaz de usuario necesaria para la gestión interna del software a desarrollar y, a su vez, el compilador y el depurador necesario para la programación del proyecto.

1.6.2. Adobe Integrated Runtime

Adobe AIR (en adelante AIR) es capaz de portar Flash Player al escritorio y a su vez a las aplicaciones ejecutándose sobre este. Otras de sus características son:

- La capacidad de crear bases de datos locales para la gestión del software en el ordenador del cliente sin necesidad alguna de comunicarse con la red.

- Sus funcionalidades más comunes son similares a las de cualquier otra aplicación instalada previamente y que utilice otra tecnología.

- Permite un manejo total del sistema de archivos, el control de los contenidos del portapapeles del sistema y la aplicación puede ser completamente actualizada de forma automática utilizando una conexión a internet. (Adobe System Incorporated 2009)

La selección del framework de AIR para el soporte de Flash Player, permite lograr, sin problema alguno, la gestión externa de un entorno de desarrollo integrado como aplicación de escritorio.

Su principal desventaja se encontraba en el hecho de no ser multiplataforma y solamente contar con una versión de prueba para Linux. En diciembre pasado, Adobe hizo pública la versión estable de AIR para Linux, eliminando esta limitante y haciéndolo completamente multiplataforma.

1.7 Lenguaje de implementación

Para la implementación de la solución se utilizará ActionScript 3.0, el lenguaje de programación de las herramientas antes mencionadas y de Adobe Flash Player. A continuación se describen las características fundamentales del mismo y que se tuvieron en cuenta a la hora de proponer la

solución existente.

1.7.1. ActionScript 3.0 (en adelante AS3):

- *Programación Orientada a Objetos*: Desde la versión 2.0 de ActionScript es posible crear proyectos Flash Player totalmente orientados a objetos. Aún así, la librería interna del framework de esa tenía la característica de ser muy inestable en tópicos como seguridad y rendimiento. AS3, creado a mediados del año 2006, es considerado una revolucionaria evolución de la versión anterior, acompañada de una serie de conceptos nuevos tomados de otros lenguajes de programación como Java y C++.

- *Alto rendimiento*: El rendimiento de AS3 es hasta 10 veces superior si se compara con sus anteriores versiones. La nueva máquina virtual para procesamiento de AS3 ("ActionScript Virtual Machine 2", en adelante AVM2), se encuentra incorporada en las versiones de Adobe Flash Player 9 y superior. De igual forma AVM1 se continúa incluyendo pese a las debilidades de las versiones anteriores frente a la nueva, debido a que, cientos de miles de contenidos en la red, continúan utilizando ActionScript 2.0.

- *Tratamiento de eventos orientado a objetos*: El tratamiento de eventos siempre ha sido una de las desventajas que los desarrolladores han tenido para crear proyectos Flash Player orientados a objetos. AS3 contiene en su framework un paquete de clases exclusivamente dedicado a la gestión de eventos de una forma muy organizada y novedosa. Es una de las características que más resaltan sus creadores.

- *Alta integración con XML*: AS3 cumple con la reciente especificación de ECMAScript para XML ("ECMAScript for XML", en adelante E4X), por lo que, nativamente, incorpora el novedoso estándar para el tratamiento de XML. E4X es otra de las características que más han propiciado la aceptación de AS3, debido a que la versión 2.0 contaba con una pobre interfaz de trabajo con el estándar de datos, lo que llevó a la creación de soluciones alternativas y a un replanteamiento del problema por parte de los creadores del lenguaje en esta nueva versión. (Grossman, Huang, 2006)

1.8 Selección de un IDE para el desarrollo

Los entornos de desarrollo de código AS3 para Flash Player no abundan. A continuación aparecen las alternativas estudiadas para la selección del mismo:

1.8.1. Flash Develop

Creado por el finlandés Mika Palmu como editor de código para MTASC y SWFMill, ha sido moderadamente utilizado por las comunidades de desarrolladores de aplicaciones para Flash Player a lo largo del mundo. Su alta difusión por la red, además de por su calidad técnica, está dada por la pobre capacidad de manejo de código que presenta el IDE de Adobe Flash, siendo una alternativa factible inclusive para los propios creadores de la herramienta de Adobe. Recientemente ha sido adaptado para soportar haXe y el SDK libre de Flex. Una de sus principales desventaja es que está desarrollado utilizando la plataforma .NET de Microsoft, por lo que solo funciona en la familia de sistemas operativos de la multinacional. (Mika Palmu, 2008)

1.8.2. Adobe Flex Builder for Linux

Adobe Flex Builder for Linux, la versión para Linux del popular entorno de desarrollo de Adobe, basado en Eclipse, aún se encuentra en un estado de pruebas que llega casi a 1 año. Comparado con la versión actual para Windows y Mac OS, la principal desventaja que tiene es que no cuenta con el entorno gráfico para la creación de las aplicaciones Flex. Aún continúa en desarrollo y no posee una versión comercial. La que actualmente se encuentra publicada a las descargas cuenta con 400 días de prueba antes de expirar. Incorpora la capacidad de crear proyectos AIR y Flex, así como depurarlos en tiempo de desarrollo y generar los proyectos finales. (Adobe System Incorporated 2008d)

Debido a las capacidades técnicas recién expuestas y a la exigencia del uso de sistemas operativos libres, Adobe Flex Builder for Linux es la mejor alternativa para desarrollar la solución planteada.

1.9 Metodologías de desarrollo

El tema de la metodología a seleccionar es uno de los más complicados en cualquier proceso de desarrollo de software, ya que son muchos los factores que se deben incluir en el análisis. A continuación se hace un estudio de las metodologías más utilizadas y que se pueden adaptar a proyectos de solo 1 persona:

1.9.1. Proceso Unificado de Desarrollo *(del inglés "Rational Unified Process", en adelante RUP)*: RUP es la metodología de desarrollo más popular que existe. Sus principales características son:

- *Guiado por Casos de uso*: Los cuales sirven para describir el comportamiento del sistema y elaborar los casos de prueba con los que se comprueba que el sistema desarrollado hace lo que el cliente quiere.

- *Centrado en la arquitectura*: La arquitectura del sistema es la columna vertebral de todo el desarrollo del mismo. Cada iteración gira en torno a la misma, fortaleciendo y corrigiendo sus características.

- *Iterativo e incremental*: En cada ciclo de iteración se produce una nueva versión del software.

Utiliza UML como lenguaje de modelado y cuenta con varias fases de trabajo en las cuales se desarrolla una serie de flujos fundamentales del desarrollo del proyecto. (Kruchten, 2002)

1.9.2. Programación Extrema *(del inglés "Extreme Programming", en adelante XP)*: Es la metodología más popular de las relativamente recientes metodologías ágiles. Sus principales características son:

- *Retroalimentación con el cliente*: Conceptualmente, al menos uno de los miembros del equipo de trabajo del proyecto, es un cliente. Esto propicia una constante interacción del mismo con el producto en desarrollo.

- *Cortas iteraciones*: En cada iteración, se obtiene un producto listo para entregar y que tiene valor para el cliente. La entrega continua de resultados compromete a ambas partes en la evolución del proyecto e indirectamente influye de forma positiva en la calidad del producto final.

- *Muy flexible a cambios*: Una de las características más reconocidas de XP. La constante retroalimentación con los clientes permite prever futuros cambios y evita llegar a momentos que paralizan el desarrollo del producto. (Shore and Warden, 2007)

Cuenta con una serie de prácticas que van en vías de aumentar la productividad del equipo de trabajo, tales como la programación en pares, reuniones diarias y planes de entrega a corto plazo, por mencionar algunos. Al igual que RUP, también utiliza UML si el equipo de desarrollo lo decide.

1.10 Selección de la metodología de desarrollo

En ambas metodologías se puede hacer un recorte amplio de roles y artefactos para adaptar el

proyecto a equipos de trabajos compuestos por solo 1 persona. RUP es conocido por la robustez de su proceso de desarrollo a largo plazo y XP por la rapidez a corto plazo de las entregas. Es debido a ello y a las cuestiones que aparecen a continuación, que XP es la metodología seleccionada para el desarrollo de la presente solución:

- El período de desarrollo es corto: El desarrollo de la solución se limita a solamente 4 meses de trabajo continuo.

- El cliente forma parte del equipo de desarrollo.

- Las dimensiones del proyecto son pequeñas.

- Uno de los objetivos específicos es publicar versiones de pruebas para obtener retroalimentación de diferentes probadores de la universidad y así optimizar el producto. Para ello es necesario tener un período de entregas corto, respaldado por iteraciones cortas y un proceso de desarrollo bastante ágil.

1.11 Herramienta CASE para el modelado

1.11.1. Visual Paradigm for UML

Como herramienta para el modelado de la solución se ha seleccionado Visual Paradigm for UML (en adelante Visual Paradigm), una de las líderes del mercado de las llamadas herramientas CASE (del inglés “Computer Asisted Software Engineering”, ingeniería de software asistida por computadora). A continuación se ofrece una lista de las características principales que se tuvieron en cuenta para la selección del mismo:

- *Soporte para la versión 2.1 de UML*: La metodología XP utiliza muchos de los diagramas que provee UML. La versión más reciente de este lenguaje es la de mayor uso a nivel mundial y la que más documentación posee.

- *Interoperabilidad entre diagramas*: Visual Paradigm es capaz de exportar los diagramas de un modelo a otro con mucha facilidad. Esto ahorra considerables cantidades de tiempo.

- *Generación de código ActionScript 3.0 desde los diagramas*. Uno de los diagramas más utilizados de UML en XP es el diagrama de clases del diseño para definir, iterativamente, la flexibilidad de la arquitectura y generar el código a partir de este. Visual Paradigm es una de las

pocas herramientas capaz de generar esta versión del lenguaje ActionScript. (Visual Paradigm 2008)

1.12 Conclusiones

En el presente capítulo se realizó un estudio sobre el estado del arte de las herramientas existentes de similar aplicación. Una de las principales conclusiones obtenidas fue que, la elección de las tecnologías utilizadas por estas, terminó siendo la principal causa de la aparición de conflictos en el momento de integración con el compilador, los cuales resultaron lamentables para la conclusión de los proyectos. La aparición de MTASC suplió la falta de un robusto compilador de ActionScript, pero solo quedó como apoyo a los proyectos del IDE de Adobe Flash.

Los recientes cambios que ha sufrido la plataforma Flash, especialmente en las licencias del SDK de Flex y la aparición de Adobe AIR, propician un ambiente ideal para combinarlos y crear un entorno de desarrollo integrado capaz de producir los componentes visuales de los software educativos y proveer a la universidad de una alternativa para revolucionar el proceso de creación de este tipo de productos en el país.

Capítulo 2

Presentación de la solución propuesta

El presente capítulo tiene como objetivo principal describir las características fundamentales del sistema. Luego se expondrán, de una manera más clara, los requerimientos a cumplir por el mismo, los cuales darán lugar a las futuras historias de usuario.

2.1 Descripción del entorno de desarrollo

La descripción inicial de cualquier solución informática es una idea que generalmente “se viene cocinando” antes de una reunión formal con un equipo de desarrollo de software. Es una visión que permite a los clientes imaginar y percibir una utilidad inmediata de uso de la misma, ya que, son ellos mismos quienes proponen esta idea a partir de sus necesidades. De ahí que sea muy común escuchar, en los primeros encuentros con estos, peticiones informales de requisitos e ideas de la interfaz de usuario, las cuales aportan mucho sobre la concepción que tienen del mismo.

XP propone, para los primeras reuniones con ellos, realizar un sencillo levantamiento de requisitos, en el cual, se tomen todos los aspectos relacionados a sus necesidades. En estos encuentros se pueden incluir prácticas muy efectivas de captura de requerimientos como:

- Entrevistas frecuentes para aclarar dudas que de manera continua puedan ir apareciendo.
- Tormenta de ideas entre el equipo de desarrollo y el equipo de trabajo del cliente.
- Observación de los procesos a informatizar manteniendo una presencia lo más imperceptible posible para no obstaculizarlos.
- Juegos de rol donde el cliente simule ser un usuario que interactúa con el sistema, este segundo simulado por el desarrollador.

En el presente trabajo el cliente es el propio desarrollador del producto. Esto podría llevar a pensar que el levantamiento de requisitos es un paso que se puede obviar. Aún en una situación así, en la que el proceso de comunicación y entendimiento de ambas partes es innecesario ya que es 1 sola persona jugando los 2 roles, sería un gran error desechar esta etapa. Atentaría totalmente contra el alcance y el tiempo de entrega del producto, dos de las cuatro variables principales en el desarrollo de un proyecto.

A continuación se especifica el alcance del producto a desarrollar de una manera similar a la que un cliente describiría. Generalmente los detalles técnicos no se incluyen en esta descripción, pero cabe aclarar que el cliente los conoce y no por esto dejan de ser requerimientos, por lo que deben ser cumplidos para lograr satisfacer sus necesidades:

Como en todo entorno de desarrollo visual, el sistema contará con un editor gráfico en el cual se podrá realizar el trabajo de creación de trazos, selección de color y edición de las figuras. A la izquierda tendrá un panel de herramientas el cual contará con las siguientes opciones de dibujo:

- Línea: Trazar una línea de un punto a otro.
- Rectángulo: Trazar un rectángulo en el área seleccionada.
- Elipse: Trazar una elipse en el área seleccionada.
- Lápiz: Trazar a mano alzada para crear figuras y trazos irregulares.
- Pluma: Trazar polígonos irregulares.
- Relleno: Selección del color de relleno de la forma de las opciones de dibujo.
- Trazo: Selección del color de trazo de la forma de las opciones de dibujo.

Todas las herramientas de dibujo funcionarán de la siguiente forma: se trazarán al hacer clic en el escenario y arrastrar el cursor por el mismo, a la vez que se muestra una vista previa de la misma antes de ser finalmente creada.

Para crear los dibujos, la herramienta centrará su funcionamiento en un framework gráfico propio y libre que permitirá a los desarrolladores utilizarlo posteriormente en otros proyectos independientemente del entorno de desarrollo en que programen.

En la parte superior del editor se mostrará el panel de propiedades en el cual se editarán las siguientes características del gráfico seleccionado:

- Relleno: Selección del color de relleno de la forma
- Trazo: Selección del color de trazo de la forma.
- Nombre de instancia: Selección del nombre de la forma para ActionScript.
- Posición horizontal (x): Valor de la posición horizontal en el eje de coordenadas (x; y).
- Posición vertical (y): Valor de la posición vertical en el eje de coordenadas (x; y).

También contará con una ventana de bienvenida que se abrirá cada vez que el usuario inicie la aplicación de manera directa (al ejecutar la aplicación desde un acceso directo). La ventana de bienvenida mostrará las siguientes secciones:

- Inicio rápido: Opción de crear un nuevo documento o abrir un documento existente.
- Documentos recientes: Ofrecerá un listado con los últimos 5 documentos previamente abiertos.

- Tutoriales: Vínculo a 4 presentaciones de instrucción on-line para los nuevos usuarios del sistema.
- Noticias recientes: Aparecerán las 5 últimas noticias del sitio WEB oficial del proyecto.
- Actualizaciones automáticas: Cada vez que sea publicada una nueva versión, los usuarios podrán actualizar su aplicación automáticamente.

El objetivo principal de esta ventana de bienvenida es ofrecer un ambiente agradable que integre el trabajo con el sistema, informaciones y noticias sobre el desarrollo del producto, una sección de instrucción para principiantes y una vía rápida de actualización a futuras versiones.

Cuando el usuario inicie la aplicación de manera indirecta (al abrir un documento creado anteriormente haciendo doble clic sobre él) no se mostrará la ventana de bienvenida y se cargará automáticamente el documento en el editor.

Una vez que los usuarios terminen el trabajo con los gráficos, tendrán la opción de abrir la ventana de generación de código ActionScript. En ella se ofrecerá la opción de exportar los gráficos a ActionScript 2.0 o 3.0, el nombre del archivo de código a exportar, la carpeta en la cual se generará el código y la vista previa del código antes de ser finalmente guardado.

Los archivos de código incluirán, a modo de comentario, ejemplos de cómo utilizarlos e incluirlos en un proyecto existente de Flash o Flex de forma tal que se puedan distribuir sin necesidad de incluir la herramienta.

En la parte superior de la aplicación aparecerá el menú de la misma con las siguientes opciones:

- Nuevo: Creará nuevo documento.
- Abrir: Abrirá un documento existente.
- Abrir reciente: Abrirá un documento utilizado recientemente.
- Salvar: Salvará el documento actual.
- Ventana de bienvenida: Mostrará la ventana de bienvenida.
- Generar código: Mostrará la ventana de generar código.
- Salir: Cerrará la aplicación.

El documento a guardar utilizará un formato abierto de fichero, de forma tal que pueda ser visualizado por otras herramientas que lo interpreten o generado a partir de otras que cumplan similar objetivo.

2.2 Personas relacionadas con el sistema

Las personas relacionadas con el sistema son todas aquellas que obtienen un resultado del mismo. Como aplicación de escritorio, un entorno de desarrollo no presenta restricciones de acceso a ciertas funcionalidades para un grupo específico de usuarios y para otros no. Todos tienen los mismos privilegios sobre las funcionalidades internas de la aplicación. Los externos, como es lógico, dependen de las restricciones clásicas que imponen los sistemas operativos, tales como el acceso a documentos de otros usuarios, las preferencias independientes para cada sesión, etc.

Personas relacionadas con el sistema	Justificación
Usuario	Puede ser tanto un diseñador gráfico sin conocimientos de programación en ActionScript como un desarrollador que utiliza la herramienta para crear componentes visuales y luego proveerlos de interactividad en Flash o Flex. En caso del usuario no tener o tener pocos conocimientos sobre el trabajo con la aplicación, puede acceder a los tutoriales de instrucción en línea para adquirir los conocimientos básicos de trabajo con la misma.

2.3 Lista de reservas del producto

La lista de reservas del producto está compuesta por una serie de requisitos que representan las tareas que el sistema debe realizar para trabajar correctamente, siempre que estas sean las que realmente el cliente quiere. Es por ello que, redactar un listado de requerimientos bien detallados y que posteriormente puedan ser probados, es tan importante.

R1: Trazar gráficos primitivos.

R1.1: Trazar línea.

R1.2: Trazar rectángulo.

R1.3: Trazar elipse.

R1.4: Realizar trazo a mano alzada.

R1.5: Realizar trazo de polígono irregular.

R1.6: Seleccionar el color de relleno de los trazos.

R1.7: Seleccionar el color de borde de los trazos.

R1.8: Mostrar la vista previa de los trazos mientras se están creando.

R2: Utilizar un framework gráfico propio para crear los gráficos.

R3: Editar las propiedades de los trazos.

R3.1: Editar el color de relleno del trazo seleccionado.

R3.2: Editar el color del borde del trazo seleccionado.

R3.3: Editar el nombre del objeto del trazo seleccionado.

R3.4: Editar la posición horizontal del trazo seleccionado en el escenario.

R3.5: Editar la posición vertical del trazo seleccionado en el escenario.

R4: Mostrar ventana de bienvenida al iniciar la aplicación.

R4.1: Crear nuevo documento.

R4.2: Abrir documento existente.

R4.3: Mostrar y abrir los 5 últimos documentos.

R4.4: Mostrar las 4 presentaciones de instrucción.

R4.5: Mostrar las 5 últimas noticias de la WEB oficial del proyecto.

R4.6: Actualizar automáticamente la aplicación a la última versión disponible.

R5: Generar código

R5.1: Generar código ActionScript 2.0 nativo.

R5.2: Generar código ActionScript 3.0 nativo.

R5.3: Generar código ActionScript 3.0 que utilice el framework propio de la aplicación.

R5.4: Seleccionar el nombre del fichero a exportar.

R5.5: Seleccionar la carpeta del fichero a exportar.

R5.6: Mostrar la vista previa del código a guardar en el fichero.

R5.7: Exportar el código final.

R6: Gestionar documento.

R6.1: Crear nuevo documento.

R6.2: Abrir documento existente.

R6.3: Abrir uno de los documentos utilizados recientemente.

R6.4: Salvar documento.

2.4 Aspectos no funcionales del sistema

Los aspectos no funcionales son propiedades o cualidades que el producto debe tener. Son características del mismo que lo hacen atractivo, usable, rápido y confiable.

Diseño e implementación

- Utilizar ActionScript 3.0 para el desarrollo del producto.
- Utilizar programación orientada a objetos.
- Aplicar patrones de diseño, principalmente para el desarrollo de editor visual y del núcleo gráfico propio. Para el núcleo gráfico es imprescindible tener en cuenta para aspectos como el rendimiento.
- Utilizar una metodología de desarrollo ágil que permita sacar versiones de manera regular.
- Utilizar Adobe AIR para garantizar la portabilidad del sistema.
- Utilizar un estándar de ficheros propio y libre para los documentos basado en XML.

Apariencia o interfaz externa

1. Utilizar componentes gráficos de Flex para el desarrollo de la interfaz de usuario.
2. Utilizar interfaces líquidas que se adapten a la resolución de pantalla del usuario.

Usabilidad

3. El sistema debe tener una versión en español y otra en inglés.
4. El sistema debe ser capaz de actualizarse automáticamente.

Seguridad

5. Verificación sobre acciones irreversibles (eliminaciones).

Ayuda y documentación en línea.

6. La documentación del núcleo gráfico debe ser realizada utilizando una herramienta generadora de documentación.

Portabilidad

7. El sistema debe ser independiente de plataforma. Debe ejecutarse tanto en Microsoft Windows como en GNU/Linux y Mac OS.

2.5 Conclusiones

En este capítulo se han abordado los aspectos referentes a la concepción del producto a construir y las características, tanto funcionales como no funcionales, que debe cumplir. Teniendo esta información, es entonces que se pasa a la fase de planificación y exploración.

Capítulo 3

Exploración y planificación

En este capítulo se hace alusión a las fases de exploración y planificación, las dos primeras de la metodología de desarrollo XP. El objetivo principal de estas es conocer el alcance del producto a desarrollar y estimar los tiempos de entrega de cada versión. Se exponen, además, los artefactos que se generan a partir de los requerimientos expuestos por el cliente en el capítulo anterior.

3.1 Exploración

La exploración es la etapa del proceso de desarrollo de software que propone XP para comenzar la construcción de un producto. Una vez que los clientes entregan su propuesta al equipo de trabajo, comienza el análisis en grupo, las horas en los pizarrones, las tormentas de ideas y la conceptualización del software. Un aspecto clave en el esclarecimiento de las dudas sobre los procesos a automatizar es que, desde el mismo inicio del proyecto, un miembro del equipo del trabajo del cliente, se adiciona al de desarrolladores.

3.2 Historias de Usuarios

Del inglés *user stories* (en adelante HU), las historias de usuarios de XP son “los hermanos lejanos” de los casos de uso en RUP, con la pequeña diferencia de que no deben ser descritos en más de tres líneas e idealmente es el cliente quien las redacta y prioriza. Luego de esto se le suma un tiempo estimado de desarrollo que lo define el propio equipo del proyecto. En esencia, no son más que las ideas del cliente organizadas y agrupadas de acuerdo a su funcionalidad. A su vez, también se tiene en cuenta un orden tal que permita al mismo priorizar sus necesidades y al equipo de trabajo definir las que resultan críticas o claves en el momento de desarrollo de la solución.

En la claridad de su descripción radica el éxito del proyecto. La falta de comunicación y la comprensión errónea de las necesidades del cliente son la principal causa de fracaso en una empresa como esta. Es por ello que las HU juegan un papel tan importante en este proceso y se les dedica totalmente una fase en el ciclo de vida de un proyecto XP. A continuación aparecen descritas las HU de la presente solución:

Historia de Usuario	
No.: 1	**Nombre:** Crear gráficos utilizando un núcleo gráfico propio
Usuario: Todos	
Prioridad en el Negocio: Alta.	**Nivel de Complejidad:** Alta.
Estimación: 1 mes (4 semanas laborales)	**Iteración Asignada:** 1
Descripción: El usuario puede crear cualquier tipo de gráfico bidimensional utilizando un framework o núcleo gráfico de ActionScript 3.0	
Información adicional (Observaciones): Da cumplimiento al requisito R2: "Utilizar un framework gráfico propio para crear los gráficos". El uso del núcleo gráfico es independiente al entorno de desarrollo. Puede ser utilizado por cualquier usuario que lo importe en un proyecto de ActionScript 3.0, Flash o Flex. El entorno de desarrollo centra su funcionamiento en él.	

Historia de Usuario	
No.: 2	**Nombre:** Gestionar documento
Usuario: Todos	
Prioridad en el Negocio: Media.	**Nivel de Complejidad:** Media.
Estimación: 2 semanas	**Iteración Asignada:** 2
Descripción: El usuario es capaz de realizar las operaciones clásicas de trabajo con un editor: abrir o cargar un documento existente, crear uno nuevo y salvar el actual. El formato del fichero a guardar será basado en XML para facilitar su entendimiento y extensibilidad.	
Información adicional (Observaciones): Da cumplimiento al requisito R6: "Gestionar documento".	

Historia de Usuario	
No.: 3	**Nombre:** Crear gráfico primitivo
Usuario: Todos	
Prioridad en el Negocio: Media.	**Nivel de Complejidad:** Media.
Estimación: 2 semanas	**Iteración Asignada:** 2
Descripción: El usuario selecciona una de los 5 posibles tipos de gráficos a crear del panel de herramientas y los traza en el escenario a la vez que se le muestra una vista previa del mismo antes de terminar el trazo.	
Información adicional (Observaciones): Da cumplimiento al requisito R1: "Trazar gráficos primitivos".	

Historia de Usuario	
No.: 4	**Nombre:** Generar código ActionScript
Usuario: Todos	
Prioridad en el Negocio: Alta.	**Nivel de Complejidad:** Media.
Estimación: 2 semanas	**Iteración Asignada:** 3
Descripción: El usuario genera código ActionScript 2.0 o 3.0 en una ubicación específica. Se le muestran opciones como utilizar código ActionScript nativo o basado en el framework propio y mostrar la vista previa del código a generar.	
Información adicional (Observaciones): Da cumplimiento al requisito R5: "Generar código".	

Historia de Usuario	
No.: 5	**Nombre:** Mostrar ventana de bienvenida al iniciar la aplicación
Usuario: Todos	
Prioridad en el Negocio: Media.	**Nivel de Complejidad:** Baja.
Estimación: 2 semanas	**Iteración Asignada:** 3
Descripción: Se le muestra al usuario la pantalla de bienvenida cada vez que inicie la aplicación desde un acceso directo. En ella tiene opciones como abrir los documentos recientes, acceso a videos de instrucción en línea, actualizaciones automáticas y notificaciones del desarrollo del producto.	
Información adicional (Observaciones): Da cumplimiento al requisito R4: "Mostrar ventana de bienvenida al iniciar la aplicación".	

Historia de Usuario	
No.: *6*	**Nombre:** Editar propiedades de los trazos
Usuario: Todos	
Prioridad en el Negocio: Baja.	**Nivel de Complejidad:** Baja.
Estimación: 2 semanas	**Iteración Asignada:** 4
Descripción: El usuario tiene la posibilidad de editar las propiedades básicas de los trazos como la posición horizontal y vertical, el nombre de la instancia y los colores de relleno y borde de la forma.	
Información adicional (Observaciones): Da cumplimiento al requisito R3: "Editar las propiedades de los trazos".	

Para la duración de las semanas que se tuvieron en cuenta en las estimaciones de las historias de usuario anteriores, es necesario aclarar que 1 semana laboral equivale a los 5 días laborales de la misma. Generalmente se hacen cálculos erróneos que estiman los tiempos de desarrollo en base a los 7 días de la semana y es válido hacer esta aclaración.

Mes de ejemplo						
Domingo	**Lunes**	**Martes**	**Miércoles**	**Jueves**	**Viernes**	**Sábado**
1	2	3	4	5	6	**7**
8	9	10	11	12	13	**14**
15	16	17	18	**19**	20	**21**
22	23	24	25	26	27	**28**
29	30	31				

	Días feriados
	Fin de semana
	Próximo mes
	Días laborales

Tabla 3.1. Análisis de los días laborales de un mes de ejemplo.

3.3 Planificación

Durante la fase de planificación se realiza una estimación del esfuerzo que costará implementar todas las historias de usuario juntas, partiendo de la que a cada una se le asignó en la fase de exploración. Una vez terminado esto, se procede a organizarlas en las iteraciones correspondientes, en dependencia de la prioridad especificada por el cliente y del tiempo de desarrollo de cada una.

3.3.1. Iteraciones

Una iteración no es más que un mini-proyecto. Al finalizar una, se obtiene un resultado en software con un valor para el cliente. Claro está que este quedará totalmente satisfecho al finalizar la última iteración, ya que es la que concluye y completa el producto acordado inicialmente.

Para organizar las iteraciones no es recomendable extenderlas en más de 1 mes laboral, lo que sería generalmente 20 o 21 días. Los plazos de entrega y retroalimentación largos atentan contra

el cumplimiento de los objetivos del cliente, sometidos a pequeños cambios de manera constante. Esto sucede, aún así, cuando un miembro de su equipo de trabajo se encuentre incluido en el de desarrollo. No hay mejor forma de evitar esto que la verificación del sistema en el entorno de despliegue del mismo. Las pruebas de los usuarios finales, las correcciones a las que se somete la aplicación en las revisiones y el análisis a partir de casos de prueba, son la mejor forma de verificar que el software avanza por el camino correcto. En resumen, alargar una iteración por más de 1 mes, es una práctica muy negativa para el desarrollo de un producto.

Iteraciones	**Orden de las Historias de usuario a implementar**	**Cantidad de tiempo de trabajo**
Iteración 1	1- Crear gráficos utilizando un núcleo gráfico propio.	4 semanas
Iteración 2	2- Gestionar documento. 3- Crear gráfico primitivo.	4 semanas
Iteración 3	4- Generar código ActionScript. 5- Mostrar ventana de bienvenida al iniciar la aplicación	4 semanas
Iteración 4	6- Editar propiedades de los trazos	2 semanas

Tabla 3.2. Historias de usuario organizadas por Iteraciones.

3.4 Plan de entregas

El plan de entregas es el compromiso final del equipo de desarrollo con los clientes. Es una cuestión de vital importancia para el negocio entre ambas partes, ya que la entrega tardía o temprana de la solución, repercute notablemente en la economía y moral de todos los involucrados. La estimación es uno de los temas más complicados del desarrollo de un proyecto de software y es por ello que resulta de vital importancia tener bien claros los requerimientos del cliente, el estilo de trabajo del equipo de desarrollo y el tiempo con que dispone el cliente para tener en sus manos la solución.

En el más extremo de los casos, la honestidad debe prevalecer en lugar de la incertidumbre y saber decir cuando se puede terminar el proyecto a tiempo o no, es muy importante. Es mejor ser claros con los clientes que defraudarlos luego de haber malgastado su tiempo y sus recursos.

Entregable	Final 1ra iteración 4º semana de Diciembre de 2008	Final 2da iteración 2º semana de Enero de 2009	Final 3ra iteración 4º Semana de Marzo de 2009	Final 4ta iteración 2º semana de Mayo de 2009
codeDraw	-	versión 0.1	versión 0.2	versión 0.5

3.5 Conclusiones

En el este capítulo se realizó la documentación de la primera etapa del ciclo de vida de la solución propuesta: los artefactos de las historias de usuario, el plan de iteraciones y el de entregas, los cuales fueron detallados de forma clara.

Capítulo 4

Implementación y pruebas

XP no propone concisamente los artefactos a utilizar en la implementación de una solución que utilice dicha metodología. Deja, en manos del equipo de desarrollo, dependiendo de sus capacidades de comunicación, la decisión de utilizar tantos tipos de diagramas de UML como crean posible, y así, facilitar el proceso de desarrollo. En el presente capítulo se hace alusión al diseño del sistema, los diagramas utilizados, las tareas generadas por cada historia de usuario y al proceso de pruebas utilizado.

4.1. Diseño del sistema

Con el objetivo de facilitar la comprensión del presente trabajo, a continuación se explican algunas características y observaciones que se tuvieron en cuenta para el desarrollo del mismo y es válida incluirlas en el inicio del capítulo.

4.1.1. El framework de Adobe Flex

Flex incorpora en su núcleo un framework desarrollado y compuesto, en su totalidad, por archivos de ActionScript 3.0 y componentes de Flash (ficheros SWC del inglés *Shockwave Component*, en adelante SWC) los cuales abarcan una amplia gama de funcionalidades. Estas van desde el acceso a servicios de redes hasta componentes de interfaz de usuario, utilidades, etc. Al crear un proyecto Flex, todas estas clases son compiladas por el propio compilador del SDK y finalmente el fichero SWF de Flash es generado.

La selección de su uso para el presente trabajo, como se explicó anteriormente en el capítulo 1, es debido a que provee la librería de componentes gráficos para el desarrollo de la interfaz visual de la solución y el compilador para trabajo con los mismos. A su vez, realiza aportes, de manera indirecta, a la concepción de los componentes visuales que se pretenden obtener en las historias de usuario Nº 1 y 4. Al Flex estar desarrollado con ActionScript para generar ActionScript, es un buen punto de referencia para la materialización del núcleo gráfico a crear.

Para el trabajo con el framework, fue seleccionado el entorno de desarrollo de sus creadores, en su versión para Linux, ya que provee un ambiente integrado muy práctico, donde convergen las librerías, el compilador y el depurador, siendo este último un factor muy importante en tiempo de desarrollo para la realización de las pruebas.

4.1.2. Adobe AIR SDK

El SDK de AIR es un compilador gratuito de Adobe para crear proyectos AIR a partir de ficheros SWF, en el caso de ActionScript, o ficheros HTML en caso de JavaScript, para luego portarlos al escritorio y proveerlos de funcionalidades que no son capaces de lograr desde un entorno WEB. AIR es una de las tecnologías más recientes de Internet y ya cuenta con una amplia documentación, comunidades de usuarios, aplicaciones muy utilizadas a lo largo del mundo y un robusto soporte.

Adobe Flex Builder permite crear proyectos Flex sobre AIR, de forma tal que el proceso de depuración es aún más rápido y no es necesario invocar al compilador del SDK de AIR para crear aplicaciones de escritorio. Flex Builder realiza todo este proceso.

4.1.3. Visual Paradigm for UML

Visual Paradigm for UML (en adelante Visual Paradigm) está orientado al trabajo con UML que, aunque no directamente fue creado para una metodología ágil, permite ser utilizado para el trabajo con artefactos importantes como son los diagramas de clases, los de colaboración, de paquetes, de despliegue, etc. (Scott W. Ambler, 2007) Adicionalmente, Visual Paradigm permite generar código ActionScript a partir de los diagramas de clases, ofreciendo una gran flexibilidad al proceso de desarrollo del núcleo y la arquitectura de la aplicación.

4.2. Descripción de la arquitectura: Aplicaciones modulares en Flex

Para el desarrollo de la interfaz de usuario se utilizó una arquitectura modular soportada sobre el patrón Modelo Vista Controlador (en adelante MVC). Flex permite crear aplicaciones modulares las cuales no son más que diferentes archivos SWF comunicándose entre sí, práctica que favorece la flexibilidad de las soluciones al no estar centralizado todo en un mismo lugar. Además permite que nuevos módulos sean integrados a la herramienta sin necesidad de modificar directamente el núcleo de la misma.

En la Figura 4.1 se muestra el diagrama de componentes de la solución, formado por los módulos de Flex. En la parte inferior del diagrama aparece *newModule*, la representación de un futuro módulo aplicable al núcleo actual:

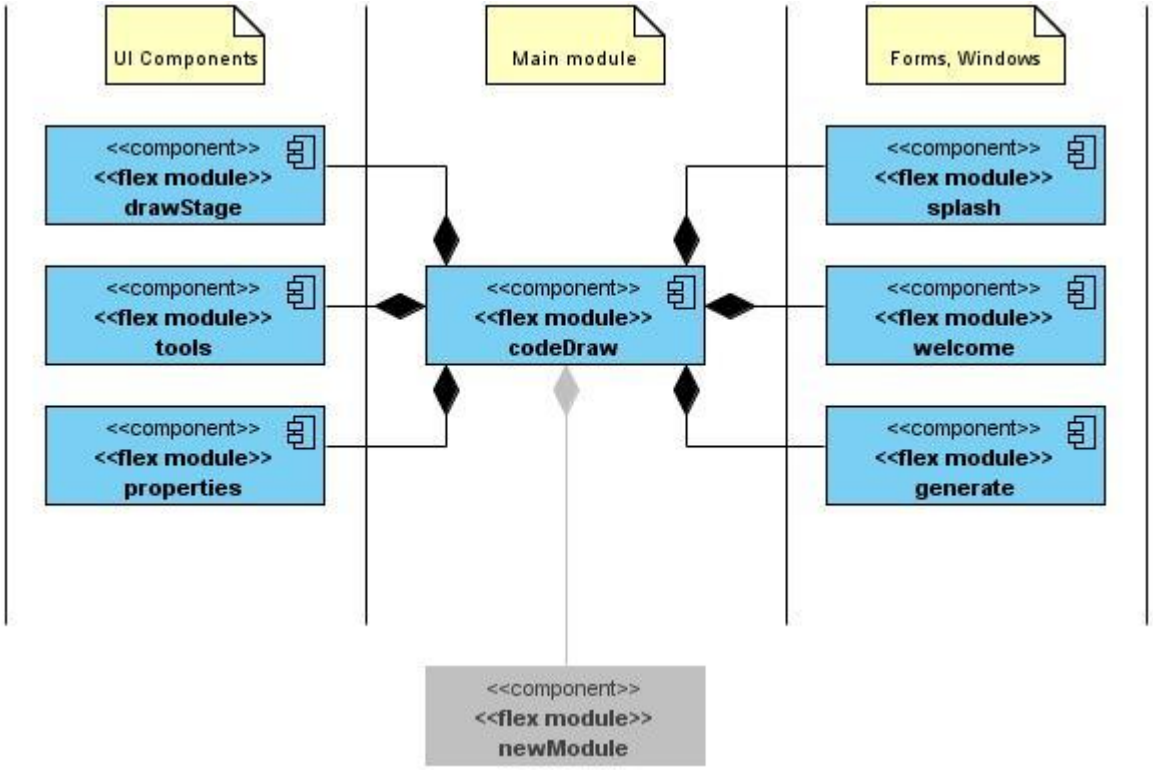

Figura 4.1. Diagrama de componentes. Módulos de Flex.

Los módulos se dividen en 3 grupos, los cuales se describen a continuación:

4.2.1. Módulo central

- **codeDraw**: Es el módulo principal de la solución. Contiene todas las instancias del MVC y a su vez es el encargado de la gestión de carga de otros módulos. Inicia el menú de la aplicación y almacena las características del documento actual.

4.2.2. Componentes de la Interfaz de usuario/Paneles

Los módulos de la interfaz de usuario de la aplicación son todos los que se integran al ambiente de desarrollo al estilo de “panel”. Ver figura 4.2 para más detalles.

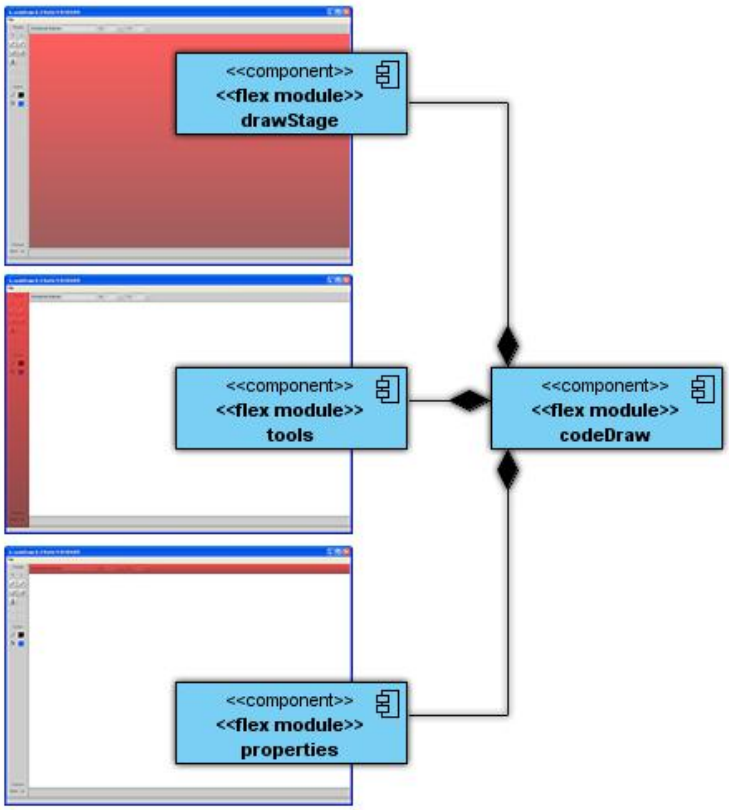

Figura 4.2. Diagrama de componentes. Componentes de la Interfaz de usuario/paneles

- **drawStage**: Módulo de dibujo. Su función principal es almacenar y visualizar los gráficos creados. Es editado por las propiedades de los otros módulos.

- **tools**: Módulo de herramientas. Contiene los componentes de interfaz de usuario, generalmente botones, encargados de definir el estilo de dibujo a utilizar, el color y algunas propiedades específicas del trazo a crear en el módulo drawStage.

- **properties**: Módulo de propiedades. Es el encargado de editar las propiedades de los gráficos existentes en el módulo drawStage.

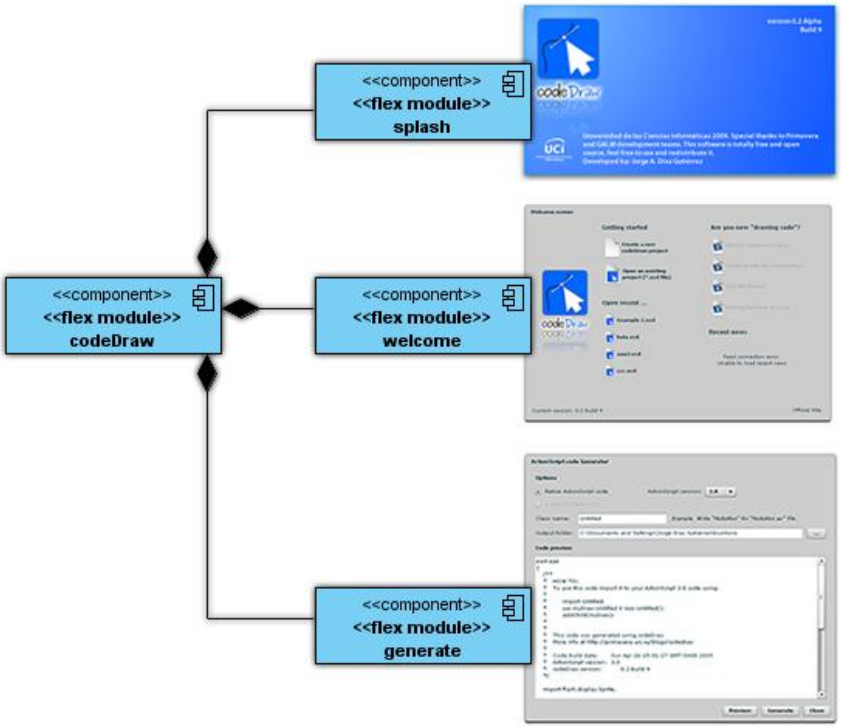

Figura 4.3. Diagrama de componentes. Ventanas/Formularios.

4.2.3. Ventanas/Formularios

- **splash**: Módulo de carga. Es el encargado de iniciar la aplicación en segundo plano, mientras muestra una pequeña ventana de inicio.

- **welcome**: Módulo de bienvenida. Contiene la lista de documentos recientes, acceso a los artículos de la Web oficial del proyecto, links a los video-tutoriales de adiestramiento y acceso a las actualizaciones automáticas.

- **generate**: Módulo de generación de código. Permite seleccionar la versión del código ActionScript a generar, mostrar la vista previa del mismo y generar el fichero final.

4.2.4. Paquetes del sistema

Para la organización del código en cada módulo, se creó una arquitectura centrada en el núcleo gráfico siguiendo una estructura convencional que se utiliza en los proyectos libres de ActionScript

y Java a lo largo del mundo. La estructura de paquetes es inversa a la que se asigna a los nombres de dominio de los sitios WEB. Por ejemplo, si se crea un proyecto para *www.nuevoSitio.cu*, la estructura de los paquetes es *cu* como paquete principal, *nuevoSitio* como secundario e internamente se organizaría el proyecto. Finalmente quedaría *cu.nuevoSitio* como el paquete raíz.

Siguiendo esta estructura, el paquete de clases de la arquitectura de la solución quedaría de la forma en que aparece en la figura 4.4. A continuación de la misma se describe cada paquete.

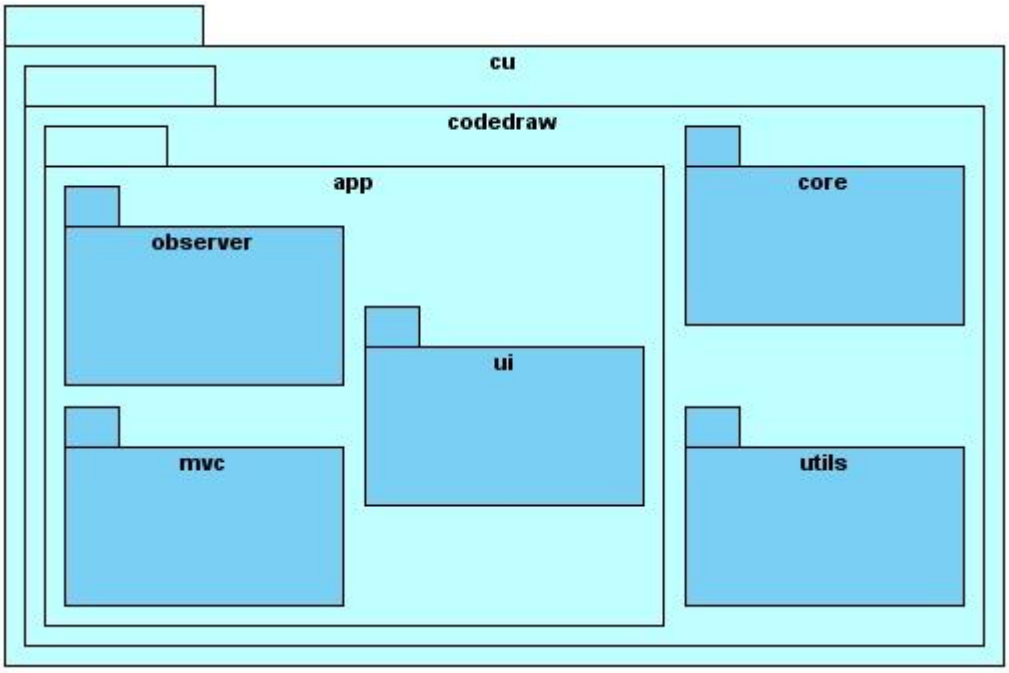

Figura 4.4. Diagrama de paquetes. Arquitectura de la solución propuesta.

core: Es el núcleo gráfico de la solución. Puede ser utilizado independientemente de los otros paquetes para la creación y edición de gráficos utilizando puro código ActionScript.

app: Es el paquete de la aplicación. Toda la gestión de la misma se realiza internamente desde este, por lo que puede ser modificado con la seguridad de que los cambios solo afectarán a la aplicación y no al núcleo gráfico.

utils: Es un paquete de clases de utilidad que permite extender las funcionalidades de la aplicación y del núcleo gráfico. No se incluyen con los mismos debido a que cumplen objetivos diferentes y pueden ser utilizadas en otros proyectos de ActionScript de similar propósito.

app.observer: Contiene la estructura del patrón Observador, el cual es clave para la

implementación del MVC. Se encuentra independiente en un paquete debido a que puede ser reutilizado sin cambio alguno.

app.mvc: Contiene la estructura del patrón Modelo Vista Controlador. Al igual que el paquete del patrón Observador, puede ser reutilizado libremente sin cambio alguno.

app.ui: Es el núcleo del paquete de la aplicación. Está compuesto concretamente por las clases que implementan el patrón MVC y definen los comportamientos de la interfaz de usuario. En su interior se ubican las clases actuales y pueden ser ubicadas las futuras extensiones de la solución.

4.2.5. Relación paquetes-módulos

Adobe Flex permite definir relaciones directas entre módulos, lo que posibilita que cada uno pueda obtener información de otro a través de interfaces previamente definidas o accediendo al módulo principal. El uso abusivo de esta relación tiende a que exista un alto acoplamiento entre todas las partes del sistema, una práctica muchas veces riesgosa y que hoy en día la principal causa de la poca flexibilidad de los proyectos creados con Flex.

Debido a esto se decidió que el módulo principal iniciara todas las instancias del MVC partiendo de su relación con los otros, limitando así, al resto de los módulos, a utilizar solamente el paquete de utilidades, con la excepción del módulo drawStage, el cual almacena la lógica del núcleo gráfico por una cuestión de rendimiento.

A continuación se muestra la relación entre los paquetes y los módulos.

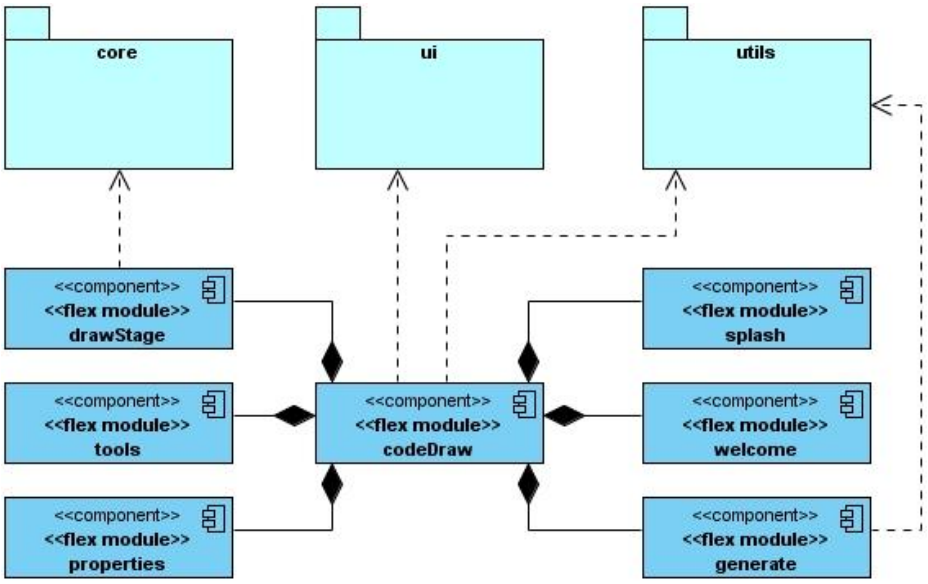

Figura 4.5. Diagrama de paquetes y módulos. Relaciones.

4.2.6. Diagrama de clases del sistema

El diagrama de clases del sistema describe la arquitectura y las relaciones entre cada una de estas clases. Es la vista más conceptual y detallada de la solución. En la Figura 4.6 se muestra el diagrama de clases de la arquitectura del sistema.

En la parte superior del mismo aparecen una serie de clases propias del lenguaje que son utilizadas para la integración de la arquitectura con el mismo:

4.2.6.1. Componentes de Interfaz de usuario de Flex

- mx.controls.*Button*
- mx.controls.*LinkButton*
- mx.controls.*Label*
- mx.controls.*ColorPicker*
- mx.core.*UIComponent*
- mx.container.*Canvas*

4.2.6.2. Núcleo de Flash Player

- flash.geom.*Matrix*
- flash,display.*Sprite*

4.2.6.3. Adobe AIR

- flash.display.*NativeWindow*
- flash.display.*NativeMenu*

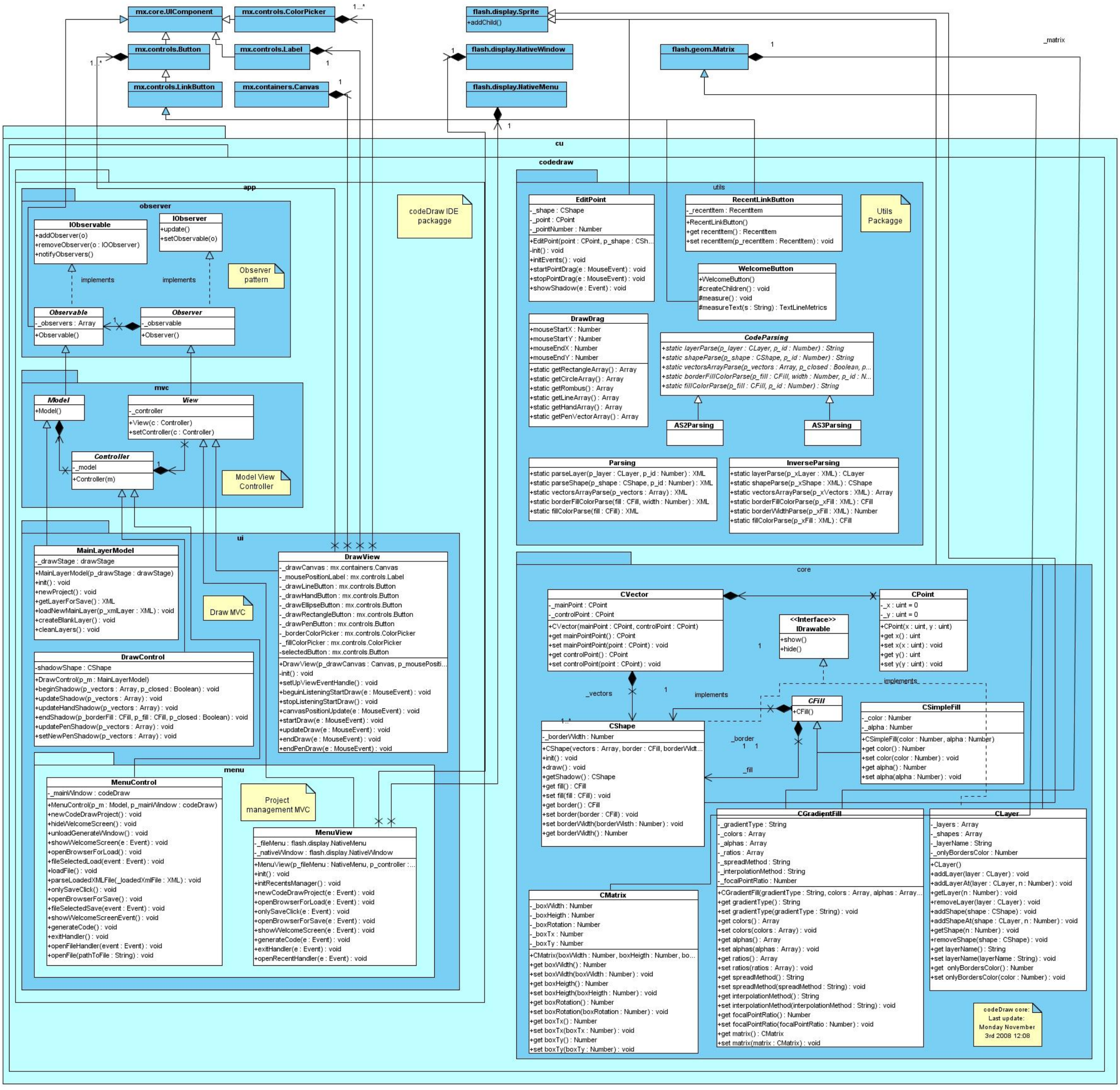

Figura 4.6. Diagrama de clases. Arquitectura de la solución propuesta.

4.2.7. Servidor de actualizaciones

El servidor de actualizaciones no es más que un servidor WEB al que se accede utilizando el protocolo HTTP. En este se encuentra la última versión disponible de la aplicación, la cual, en caso de ser superior a la que se encuentra en el ordenador del cliente, se instala automáticamente una vez que los usuarios hacen clic en la notificación.

De igual forma se accede al servidor WEB que contiene el Blog del proyecto. Utilizando el canal de noticias del mismo, en formato RSS 2.0, se descargan y son visualizadas en el módulo de bienvenida, de forma tal que este ofrezca un ambiente integrado de notificaciones y artículos de la WEB oficial.

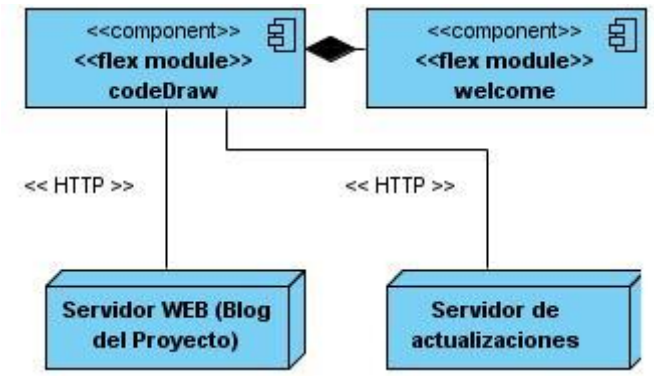

Figura 4.7. Diagrama de despliegue. Distribución de servidores que se integran a la solución.

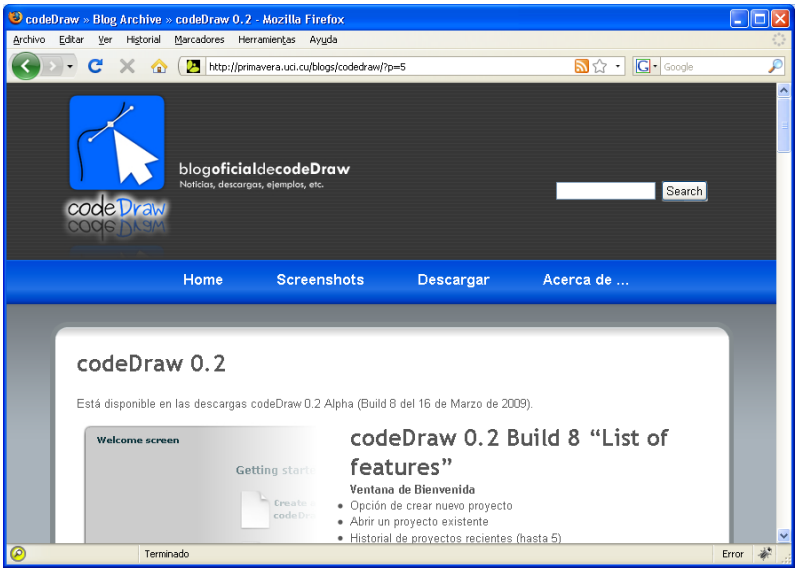

Figura 4.8. Captura de pantalla. WEB Oficial del Proyecto.

4.3. Implementación

XP propone comenzar las implementación de la solución partiendo de una arquitectura lo más flexible posible para evitar grandes cambios en las próximas iteraciones y en los cambios que generalmente el cliente propone. Como se ha mencionado otras veces en el presente trabajo, la solución en cuestión cuenta con el detalle que es de carácter técnico, por lo que es necesario, desde un inicio, tener bien definida la arquitectura del mismo.

Trazada la misma, se procede a la primera iteración.

4.3.1. 1º Iteración

El principal objetivo de la primera iteración es desarrollar la historia de usuario Nº 1: Crear gráficos utilizando un núcleo gráfico propio. Esta iteración no finaliza con un producto visual capaz de ser probado, ya que su resultado es una librería gráfica en forma de código y servir de base para las próximas iteraciones.

Para ello se trazaron 4 tareas, que se describen a continuación:

1. Tarea Nº 1: Estándar de código y de comentarios del núcleo
2. Tarea Nº 2: Clases principales del núcleo: Clases Matemáticas (Nivel 1)
3. Tarea Nº 3: Arquitectura de capas
4. Tarea Nº 4: Clases de graficado: Clases de trazado y relleno (Nivel 2)

Tarea	
Número de tarea: 1	**Número de HU**: 1
Nombre de la tarea: Estándar de código y de comentarios	
Tipo de tarea: Configuración	**Estimación**: 3 días
Fecha inicio:11 de Noviembre de 2008	**Fecha fin**: 14 de Noviembre de 2008
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Redactar el estándar de código a utilizar siguiendo los utilizados internacionalmente en proyectos de ActionScript 3.0. Consultar documentación en la red para la definición del mismo.	

Tarea	
Número de tarea: 2	**Número de HU**: 1
Nombre de la tarea: Clases principales del núcleo: Clases Matemáticas	
Tipo de tarea: Desarrollo	**Estimación**:5 dias
Fecha inicio:15 de Noviembre de 2008	**Fecha fin**: 20 de Noviembre de 2008
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Implementar las clases principales del núcleo. **Paquete**: *cu.codeDraw.core* **Clases**: *CPoint, CVector, CMatrix*	

Tarea	
Número de tarea: 3	**Número de HU**: 1
Nombre de la tarea: Arquitectura de capas	
Tipo de tarea: Desarrollo	**Estimación**: 5 días
Fecha inicio: 21 de Noviembre de 2008	**Fecha fin**: 26 de Noviembre de 2008
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Desarrollar la arquitectura de capas que permita gestionar los grupos de imágenes por encima de otros e independientemente a sus características. **Paquete**: *cu.codeDraw.core* **Clases**: *CLayer, IDrawable*	

Tarea	
Número de tarea: 4	**Número de HU**: 1
Nombre de la tarea: Clases de graficado: Clases de trazado y relleno	
Tipo de tarea: Desarrollo	**Estimación**: 7 días
Fecha inicio: 27 de Noviembre de 2008	**Fecha fin**: 5 de Diciembre del 2008
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Desarrollo de las clases de graficado, las encargadas de la creación, edición y almacenamiento de los componentes gráficos. **Paquete**: *cu.codeDraw.core* **Clases**: *CShape, CFill, CSimpleFill, CGradientFill*	

4.3.2. 2ª Iteración

La segunda iteración persigue el objetivo de obtener la primera versión de un producto que cumpla con las historias de usuario Nº 2 y 3. Estas son “Gestionar documento” y “Crear gráfico primitivo” respectivamente. Una vez creadas las bases de la gestión de ficheros y de trazado de gráficos, el núcleo de la aplicación visual está terminado para crear tantas extensiones como sea necesario.

Las tareas definidas para la iteración son las siguientes:

Historia de Usuario Nº 2 “Gestionar documento”.

1. Tarea Nº 5: Estándar XML de los proyectos
2. Tarea Nº 6: Interfaz de usuario Nuevo, Abrir, Cerrar y Guardar
3. Tarea Nº 7: Parseo del núcleo del Proyecto gráfico
4. Tarea Nº 8: Nuevo, Abrir, Cerrar y Guardar

Historia de Usuario Nº 3 “Crear gráfico primitivo”.

5. Tarea Nº 9: Interfaz de usuario Línea, Rectángulo, Elipse, Lápiz y Pluma
6. Tarea Nº 10: Sombra previa del gráfico a trazar
7. Tarea Nº 11: Trazar el gráfico.

Tarea	
Número de tarea: 5	**Número de HU**: 2
Nombre de la tarea: Estándar XML de los proyectos	
Tipo de tarea: Configuración	**Estimación**:1 dia
Fecha inicio:19 de Enero de 2009	**Fecha fin**: 19 de Enero de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Redactar el estándar XML de los ficheros del proyecto. Crear los íconos de los ficheros XML de codeDraw (*.xcd)	

Tarea	
Número de tarea: 6	**Número de HU**: 2
Nombre de la tarea: Interfaz de usuario Nuevo, Abrir, Cerrar y Guardar	
Tipo de tarea: Desarrollo	**Estimación**:1 dia
Fecha inicio: 20 de Enero de 2009	**Fecha fin**: 20 de Enero de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Diseño de la interfaz de usuario. **Módulo**: *codeDraw*.	

Tarea	
Número de tarea: 7	**Número de HU**: 2
Nombre de la tarea: Parseo del núcleo del Proyecto gráfico	
Tipo de tarea: Desarrollo	**Estimación**: 2 días
Fecha inicio: 21 de Enero de 2009	**Fecha fin**: 22 de Enero de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Desarrollar las clases de parseo del núcleo gráfico y de los ficheros xcd. **Paquete**: *cu.codeDraw.utils* **Clases**: *Parsing, InverseParsing*	

Tarea	
Número de tarea: 8	**Número de HU**: 2
Nombre de la tarea: Nuevo, Abrir, Cerrar y Guardar	
Tipo de tarea: Desarrollo	**Estimación**: 6 días
Fecha inicio: 23 de Enero de 2009	**Fecha fin**: 30 de Enero de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Implementación de las funcionalidades de cargar un proyecto previamente salvado, salvar el documento actual, crear nuevo documento y cerrar el actual.. **Paquete**: *cu.codeDraw.app, cu.codeDraw.app.ui* **Clases**: *MainLayerModel, MenuControl, MenuView* **Módulo**: *codeDraw*	

Tarea	
Número de tarea: 9	**Número de HU**: 3
Nombre de la tarea: Interfaz de usuario Línea, Rectángulo, Elipse, Lápiz y Pluma	
Tipo de tarea: Desarrollo	**Estimación**:1 dia
Fecha inicio:16 de Febrero de 2009	**Fecha fin**: 16 de Febrero de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Diseño de la interfaz de usuario. **Módulo**: *codeDraw, tools*	

Tarea	
Número de tarea: 10	**Número de HU**: 3
Nombre de la tarea: Sombra previa del gráfico a trazar	
Tipo de tarea: Desarrollo	**Estimación**: 2 días
Fecha inicio: 17 de Febrero de 2009	**Fecha fin**: 18 de Febrero de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Implementar la vista previa de los trazos a medida que el usuario se desplaza por el escenario. **Paquete**: *cu.codeDraw.utils, cu.codeDraw.app.ui* **Clase**: *DrawDrag, DrawView, DrawControl* **Módulo**: *codeDraw, drawStage*	

Tarea	
Número de tarea: 11	**Número de HU**: 3
Nombre de la tarea: Trazar el gráfico	
Tipo de tarea: Desarrollo	**Estimación**: 7 días
Fecha inicio: 18 de Febrero de 2009	**Fecha fin**: 26 de Febrero de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Desarrollar el trazo de los gráficos primitivos. **Paquete**: *cu.codeDraw.utils, cu.codeDraw.app.ui* **Clase**: *DrawDrag, DrawView, DrawControl* **Módulo**: *drawStage*	

4.3.3. 3ª Iteración

En la tercera iteración se logra el objetivo principal de la solución: generar el código ActionScript de los componentes visuales. Las historias de usuario a cumplir son “Generar código ActionScript” y “Mostrar ventana de bienvenida al iniciar la aplicación”, la segunda para lograr un ambiente integrado de la aplicación con la WEB del proyecto y la gestión de documentos recientes.

Historia de Usuario Nº 4 “Gestionar documento”.

1. Tarea Nº 12: Estándar de código generado
2. Tarea Nº 13: Parseo del núcleo del proyecto gráfico
3. Tarea Nº 14: Interfaz de usuario generar código
4. Tarea Nº 15: Generar código ActionScript 2.0
5. Tarea Nº 16: Generar código ActionScript 3.0
6. Tarea Nº 17: Mostrar vista previa y salvar fichero

Historia de Usuario Nº 5 “Mostrar ventana de bienvenida al iniciar la aplicación”.

7. Tarea Nº 18: Interfaz de usuario de la ventana de bienvenida y de carga
8. Tarea Nº 19: Abrir archivos recientes
9. Tarea Nº 20: Últimas noticias del Blog
10. Tarea Nº 21: Actualizaciones automáticas

Tarea	
Número de tarea: 12	**Número de HU**: 4
Nombre de la tarea: Estándar de código generado	
Tipo de tarea: Configuración	**Estimación**:1 dia
Fecha inicio: 2 de Marzo de 2009	**Fecha fin**: 2 de Marzo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Redactar el estándar de codificación siguiendo los utilizados en la creación de la solución. Introducir un encabezado de comentarios con información sobre el código y cómo utilizarlo.	

Tarea	
Número de tarea: 13	**Número de HU**: 4
Nombre de la tarea: Parseo del núcleo del proyecto gráfico	
Tipo de tarea: Desarrollo	**Estimación**: 1 dia
Fecha inicio: 3 de Marzo de 2009	**Fecha fin**: 3 de Marzo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción:	
Parseo del núcleo gráfico para generar el código ActionScript.	
Paquete: *cu.codeDraw.utils*	
Clases: *CodeParsing*	

Tarea	
Número de tarea: 14	**Número de HU**: 4
Nombre de la tarea: Interfaz de usuario generar código	
Tipo de tarea: Desarrollo	**Estimación**: 1 días
Fecha inicio: 4 de Marzo de 2009	**Fecha fin**: 4 de Marzo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción:	
Diseño de la interfaz de usuario de la ventana de generar código.	
Módulo: *generate*	

Tarea	
Número de tarea: 15	**Número de HU**: 4
Nombre de la tarea: Generar código ActionScript 2.0	
Tipo de tarea: Desarrollo	**Estimación**: 4 días
Fecha inicio: 5 de Marzo de 2009	**Fecha fin**: 10 de Marzo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción:	
Generar el código ActionScript 2.0 de los gráficos creados en el escenario.	
Paquete: *cu.codeDraw.utils,*	
Clase: *AS2Parsing*	

Tarea	
Número de tarea: 16	**Número de HU**: 4
Nombre de la tarea: Generar código ActionScript 3.0	
Tipo de tarea: Desarrollo	**Estimación**:1 dia
Fecha inicio:11 de Marzo de 2009	**Fecha fin**: 11 de Marzo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Generar el código ActionScript 3.0 de los gráficos creados en el escenario. **Paquete**: *cu.codeDraw.utils,* **Clase**: *AS3Parsing*	

Tarea	
Número de tarea: 17	**Número de HU**: 4
Nombre de la tarea: Mostrar vista previa y salvar fichero	
Tipo de tarea: Desarrollo	**Estimación**: 2 días
Fecha inicio: 12 de Marzo de 2009	**Fecha fin**: 13 de Marzo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Implementar la vista previa de los trazos a medida que el usuario se desplaza por el escenario. **Módulo**: *generate*	

Tarea	
Número de tarea: 18	**Número de HU**: 5
Nombre de la tarea: Interfaz de usuario de la ventana de bienvenida y de carga	
Tipo de tarea: Desarrollo	**Estimación**: 2 días
Fecha inicio: 16 de Marzo de 2009	**Fecha fin**: 17 de Marzo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Diseñar la interfaz de usuario de la ventana de bienvenida y de la ventana de carga de la aplicación. **Módulo**: *welcome, splash*	

Tarea	
Número de tarea: 19	**Número de HU**: 5
Nombre de la tarea: Abrir archivos recientes	
Tipo de tarea: Desarrollo	**Estimación**:3 dia
Fecha inicio: 17 de Marzo de 2009	**Fecha fin**: 20 de Marzo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: **Paquete**: *cu.codeDraw.app.ui* **Clases**: *MenuControl, MenuView* **Módulo**: *codeDraw, welcome*	

Tarea	
Número de tarea: 20	**Número de HU**: 5
Nombre de la tarea: Últimas noticias del Blog	
Tipo de tarea: Desarrollo	**Estimación**: 3 días
Fecha inicio: 23 de Marzo de 2009	**Fecha fin**: 25 de Marzo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Implementar el acceso al canal de sindicación de noticias del Blog para visualizarlas. **Módulo**: *welcome*	

Tarea	
Número de tarea: 21	**Número de HU**: 5
Nombre de la tarea: Actualizaciones automáticas	
Tipo de tarea: Desarrollo	**Estimación**: 2 días
Fecha inicio: 26 de Marzo de 2009	**Fecha fin**: 27 de Marzo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Implementar un sistema de actualizaciones automáticas desde el Servidor de actualizaciones. **Módulo**: *welcome*	

4.3.4. 4ª Iteración

La cuarta y última iteración tiene como propósito ofrecer a los usuarios opciones básicas de edición al cumplir la historia de usuario Nº 6 “Editar propiedades de los trazos”. Las tareas para darle cumplimiento a la misma son las siguientes:

1. Tarea Nº 22: Editar la posición y nombre de la instancia
2. Tarea Nº 23: Editar colores del trazo

Tarea	
Número de tarea: 22	**Número de HU**: 5
Nombre de la tarea: Editar la posición y nombre de la instancia	
Tipo de tarea: Desarrollo	**Estimación**: 5 días
Fecha inicio: 3 de Mayo de 2009	**Fecha fin**: 8 de Mayo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Implementar el acceso al canal de sindicación de noticias del Blog para visualizarlas. **Paquete**: *cu.codeDraw.utils* **Clases**: *EditPoint* **Módulo**: *drawStage, tools*	

Tarea	
Número de tarea: 23	**Número de HU**: 5
Nombre de la tarea: Editar colores del trazo	
Tipo de tarea: Desarrollo	**Estimación**: 5 días
Fecha inicio: 10 de Mayo de 2009	**Fecha fin**: 14 de Mayo de 2009
Programador responsable: Jorge Antonio Díaz Gutiérrez	
Descripción: Implementar un sistema de actualizaciones automáticas desde el Servidor de actualizaciones. **Módulo**: *tools, drawStage*	

4.4. Pruebas

XP cuenta con una característica conocida como TDD, desarrollo dirigido por pruebas (del inglés *test driven development*).

TDD se enfoca en la implementación orientada a pruebas. El código debe ser probado paso a paso y obtener un resultado, aunque aún no con lógica para el negocio, sí funcional. Puede confundirse con el término “pruebas de caja blanca” y, hasta cierto punto, es así:

Las pruebas de caja blanca son realizadas a los métodos u operaciones para medir la funcionalidad del mismo desde la perspectiva de la validez para el cliente. TDD, por otro lado, se aplica antes de comenzar a implementar cada paso de la tarea en desarrollo asumiendo que la prueba es insatisfactoria desde un inicio. Solo, una vez que se haya cumplido de la forma más sencilla posible, la lógica del código a probar, se asume como cumplida. Luego se realiza un proceso conocido informalmente como “refactorización” de código, el cual consiste en limpiarlo, organizarlo y adaptarlo a los patrones. En esencia, TDD se centra en la lógica del código y las pruebas de caja blanca en la del negocio. Las pruebas TDD también son conocidas como pruebas unitarias o de unidad.

ActionScript no cuenta con un framework de pruebas TDD, por lo que es necesario utilizar el depurador de Flex Builder o la consola de salida en última instancia.

Una vez terminada cada tarea, se prueba la funcionalidad de la misma, ya desde el punto de la validez de esta para el cliente. Esta prueba es realizada por parte del desarrollador. Luego se realizan las pruebas conocidas como de “caja gris”, en la cual se valida el comportamiento general de la solución en ambientes adversos como la pérdida de conectividad con los servicios de redes, pocos recursos de hardware y sobrecarga del mismo. Finalmente el usuario final, realiza las de “caja negra” o aceptación.

4.4.1. Pruebas de aceptación

Las pruebas de aceptación también son conocidas como pruebas de caja negra, ya que es el propio cliente quien la realiza en compañía de uno de los representantes del equipo de desarrollo y se orientan a las funcionalidades del sistema. Su objetivo es comprobar, desde la perspectiva del usuario final, el cumplimiento de las especificaciones de la lista de reservas del producto.

A continuación, aparecen las pruebas de aceptación realizadas a la solución propuesta:

Caso de Prueba de Aceptación	
Código: HU1_P1	**Historia de Usuario**: 1
Nombre: Crear gráfico poligonal recto	
Descripción: Prueba para la funcionalidad de crear gráficos rectos utilizando el núcleo del sistema.	
Condiciones de Ejecución: El usuario debe tener creado un proyecto en Flex Builder.	
Entrada/ Pasos de ejecución: Se importan las clases del núcleo gráfico y se genera una serie de puntos aleatorios para la creación de un polígono recto. Se crea un nuevo objeto gráfico con los puntos. Luego se compila el código.	
Resultado Esperado: El gráfico es creado y graficado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU1_P2	**Historia de Usuario**: 1
Nombre: Crear gráfico poligonal curvo	
Descripción: Prueba para la funcionalidad de crear gráficos curvos utilizando el núcleo del sistema.	
Condiciones de Ejecución: El usuario debe tener creado un proyecto en Flex Builder.	
Entrada/ Pasos de ejecución: Se importan las clases del núcleo gráfico y se genera una serie de puntos aleatorios para la creación de un polígono curvo. Se crea un nuevo objeto gráfico con los puntos. Luego se compila el código.	
Resultado Esperado: El gráfico es creado y graficado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU2_P1	**Historia de Usuario**: 2
Nombre: Crear nuevo documento	
Descripción: Prueba para la funcionalidad de crear nuevo documento XCD.	
Condiciones de Ejecución: -	
Entrada/ Pasos de ejecución: Se selecciona crear nuevo documento.	
Resultado Esperado: El documento es creado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU2_P2	**Historia de Usuario**: 2
Nombre: Guardar documento existente	
Descripción: Prueba para la funcionalidad de salvar un documento XCD.	
Condiciones de Ejecución: El usuario debe haber creado previamente un documento.	
Entrada/ Pasos de ejecución: Se selecciona guardar el documento actual, navega a la ubicación deseada, escribe el nombre del nuevo documento y hace clic en salvar.	
Resultado Esperado: El documento es guardado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU2_P3	**Historia de Usuario**: 2
Nombre: Abrir documento existente	
Descripción: Prueba para la funcionalidad de cargar un documento XCD salvado previamente.	
Condiciones de Ejecución: Debe existir un documento XCD previamente creado.	
Entrada/ Pasos de ejecución: Se selecciona abrir documento, se navega hacia la nueva ubicación, se selecciona el documento y se hace clic en salvar.	
Resultado Esperado: El documento es cargado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de Usuario**: 3
Nombre: Crear línea	
Descripción: Prueba para la funcionalidad de crear una nueva línea.	
Condiciones de Ejecución: El usuario debe haber creado previamente un documento.	
Entrada/ Pasos de ejecución: Se selecciona crear nueva línea y se traza en el escenario.	
Resultado Esperado: El gráfico es creado y graficado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU3_P2	**Historia de Usuario**: 3
Nombre: Crear elipse	
Descripción: Prueba para la funcionalidad de crear una nueva elipse.	
Condiciones de Ejecución: El usuario debe haber creado previamente un documento.	
Entrada/ Pasos de ejecución: Se selecciona crear nueva elipse y se traza en el escenario.	
Resultado Esperado: El gráfico es creado y graficado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU3_P3	**Historia de Usuario**: 3
Nombre: Crear rectángulo	
Descripción: Prueba para la funcionalidad de crear un nuevo rectángulo.	
Condiciones de Ejecución: El usuario debe haber creado previamente un documento.	
Entrada/ Pasos de ejecución: Se selecciona crear nuevo rectángulo y se traza en el escenario.	
Resultado Esperado: El gráfico es creado y graficado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU3_P4	**Historia de Usuario**: 3
Nombre: Crear polígono irregular	
Descripción: Prueba para la funcionalidad de crear un nuevo polígono irregular.	
Condiciones de Ejecución: El usuario debe haber creado previamente un documento.	
Entrada/ Pasos de ejecución: Se selecciona crear nuevo polígono irregular y se traza en el escenario punto a punto, luego se cierra con clic derecho.	
Resultado Esperado: El gráfico es creado y graficado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU3_P5	**Historia de Usuario**: 3
Nombre: Crear trazo a mano alzada	
Descripción: Prueba para la funcionalidad de crear un nuevo trazo a mano alzada.	
Condiciones de Ejecución: El usuario debe haber creado previamente un documento.	
Entrada/ Pasos de ejecución: Se selecciona crear nuevo trazo a mano alzada y se traza en el escenario.	
Resultado Esperado: El gráfico es creado y graficado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU4_P1	**Historia de Usuario**: 4
Nombre: Generar código ActionScript 3.0	
Descripción: Prueba para la funcionalidad de generar código ActionScript 3.0	
Condiciones de Ejecución: El usuario debe haber creado previamente un documento.	
Entrada/ Pasos de ejecución: Se selecciona generar código ActionScript, se selecciona la versión 3.0, se define el nombre del fichero a generar y la ubicación y luego se genera.	
Resultado Esperado: El código es generado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU4_P2	**Historia de Usuario**: 4
Nombre: Generar código ActionScript 2.0	
Descripción: Prueba para la funcionalidad de generar código ActionScript 3.0	
Condiciones de Ejecución: El usuario debe haber creado previamente un documento.	
Entrada/ Pasos de ejecución: Se selecciona generar código ActionScript, se selecciona la versión 2.0, se define el nombre del fichero a generar y la ubicación y luego se genera.	
Resultado Esperado: El código es generado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU4_P3	**Historia de Usuario**: 4
Nombre: Mostrar vista previa	
Descripción: Prueba para la funcionalidad mostrar la vista previa del código a generar	
Condiciones de Ejecución: El usuario debe haber creado previamente un documento.	
Entrada/ Pasos de ejecución: Se selecciona generar código ActionScript, se selecciona la versión del lenguaje, se define el nombre del fichero a generar y luego se muestra la vista previa.	
Resultado Esperado: El código es pre-visualizado sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU5_P1	**Historia de Usuario**: 5
Nombre: Actualizar la aplicación	
Descripción: Prueba para la funcionalidad de actualizar la aplicación.	
Condiciones de Ejecución: -	
Entrada/ Pasos de ejecución: Se hace clic en Actualizar a última versión.	
Resultado Esperado: La herramienta se actualiza automáticamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU5_P2	**Historia de Usuario**: 5
Nombre: Mostrar últimas noticias	
Descripción: Prueba para la funcionalidad mostrar últimas noticias del Blog.	
Condiciones de Ejecución: -	
Entrada/ Pasos de ejecución: El usuario accede a las noticias desde la pantalla de Bienvenida, las cuales se cargan automáticamente.	
Resultado Esperado: Las noticias se muestran automáticamente.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU5_P3	**Historia de Usuario**: 5
Nombre: Abrir documentos recientes	
Descripción: Prueba para la funcionalidad de abrir documentos recientes.	
Condiciones de Ejecución: Deben haberse abierto o salvado, al menos 1 documento previamente.	
Entrada/ Pasos de ejecución: Se selecciona el documento reciente a abrir.	
Resultado Esperado: El documento se abre sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU6_P1	**Historia de Usuario**: 6
Nombre: Editar la posición del gráfico	
Descripción: Prueba para la funcionalidad de editar las propiedades de ubicación de los gráficos.	
Condiciones de Ejecución: El usuario debe tener gráficos previamente cargados o creados en el escenario.	
Entrada/ Pasos de ejecución: El usuario selecciona el gráfico a editar y modifica su posición desde el panel de propiedades o arrastrándolo hacia una nueva posición.	
Resultado Esperado: La posición se modifica sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

Caso de Prueba de Aceptación	
Código: HU6_P2	**Historia de Usuario**: 6
Nombre: Editar los colores del gráfico.	
Descripción: Prueba para la funcionalidad de editar los colores de los gráficos.	
Condiciones de Ejecución: El usuario debe tener gráficos previamente cargados o creados en el escenario.	
Entrada/ Pasos de ejecución: El usuario selecciona el gráfico a editar y modifica sus colores desde el panel de propiedades.	
Resultado Esperado: Los colores se modifican sin errores.	
Evaluación de la Prueba: Prueba satisfactoria.	

4.5. Conclusiones

En el presente capítulo se abordaron todos los temas referentes a la implementación de la solución y a la estrategia de pruebas a seguir durante la elaboración del mismo. Se presentó el diseño de la arquitectura y las tareas que se llevaron a cabo para construirla.

Cada tarea fue cumplida en fecha y finalmente se logró el producto deseado en el plazo de tiempo planificado. La efectividad del desarrollo dirigido por pruebas y la aplicación de las pruebas de aceptación demuestran ser muy alta. Ambas juegan un papel fundamental en el proceso de desarrollo de un software con una metodología ágil, aún cuando el equipo de desarrollo es de 1 solo integrante.

Conclusiones

Una vez finalizado el desarrollo del presente trabajo, se puede concluir con la satisfacción de haber obtenido una herramienta capaz de crear sencillos componentes visuales y generar el código ActionScript de los mismos. La capacidad de extensión de la solución, para lograr objetivos adicionales, le permite ser la base robusta de un ambiente de desarrollo extensible y fácil de continuar construyendo. Ese ha sido uno de los principios fundamentales seguidos en el proceso de desarrollo del mismo. A su vez, el uso de la metodología de desarrollo XP, combinada con los aportes arquitectónicos que provee UML, ha permitido desarrollar, en un plazo corto de tiempo, un producto bastante maduro.

Una de las conclusiones principales es que ActionScript 3.0 es un lenguaje de programación muy robusto para crear contenidos interactivos, ya sean de software educativo u otro tipo de aplicaciones de alto impacto en el usuario final. El principio de generar código para luego ser compilado con el SDK libre de Adobe Flex, es fundamental para crear alternativas legales de explotar la tecnología y conservar las capacidades de la misma, en lugar de optar por compiladores que se limitan a funcionalidades más reducidas. A su vez evita crear complejos intérpretes que persiguen cumplir un objetivo similar, pero se convierten en soluciones muy poco flexibles. Ante una futura evolución del lenguaje y de la plataforma, los generadores de código se adaptan de forma rápida, ya que los cambios que sufren son relativamente reducidos.

Recomendaciones

La solución desarrollada en el presente trabajo ha permitido llegar a una serie de conclusiones muy importantes para la continuidad del desarrollo de la misma y, a su vez, sirven de experiencias para proyectos que persiguen objetivos similares, hasta cierto punto:

- **Usabilidad**: Los ambientes de desarrollo deben ser capaces de integrar, de manera subjetiva, a las comunidades de desarrolladores y tener, muy en cuenta, las recomendaciones de los mismos para lograr el impacto realmente esperado y evitar futuros rechazos. Un IDE es una herramienta que necesita ser utilizada de manera frecuente, por lo que la usabilidad es uno de los paradigmas que deben guiar su producción. La presente solución podría categorizarse como aceptable, pero aún no práctica para un usuario familiarizado con ambientes de desarrollo más populares.

- **Nuevas funcionalidades**: Para las próximas versiones incorporar texto, una librería de formas prediseñadas, filtros, gestión de capas, opciones de edición de puntos, selección múltiple, historial de acciones y portapapeles de copia.

- **Animación**: Las capacidades de animación que puede asumir el producto no cuentan con límites de tecnología. Si sobre Flash Player y AIR es posible desarrollar una herramienta de este tipo, se puede afirmar que la extensión de la misma en esta dirección dependerá siempre de enfoques óptimos, en términos de rendimiento, ya que es en este aspecto donde se encuentra la mayor debilidad de la plataforma de Adobe.

- **Gestión de eventos**: Incorporar una sencilla gestión de eventos que permita generar el código de las funcionalidades básicas de los componentes.

Glosario

Adobe System Incorporated: Empresa norteamericana de software con sede en California. Fue fundada en diciembre de 1982 por John Warnock y Charles Geschke. Destaca en el mundo del software por sus programas de edición de páginas web, vídeo e imagen digital.

AVM: Máquina Virtual de ActionScript (del inglés “ActionScript Virtual Machine”) es la parte de Adobe Flash Player que se encarga de procesar los códigos de ActionScript cuando se está ejecutando una película de un archivo SWF.

Compilador: Un compilador es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar. Usualmente el segundo lenguaje es código máquina, pero también puede ser simplemente texto.

Framework: Un framework, en el desarrollo de software, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, librerías de código y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Macromedia: Macromedia era una empresa norteamericana de software de gráficos y desarrollo web con centrales en California. Macromedia fue formada en 1992 por la fusión de Authorware, Inc. y MacroMind-Paracomp. En el año 2005 fue absorbida por la empresa Adobe Systems Incorporated, por lo cual todos sus productos y tecnologías comenzaron a salir al mercado bajo un nuevo nombre.

On-line: Concepto de no más de 20 años de aparición que significa “en línea”. Es una categoría que se le da a las aplicaciones o procesos que se ejecutan a partir del uso de una computadora conectada a la red de redes.

Plataforma .NET de Microsoft: Plataforma de la empresa Microsoft para el desarrollo de software con énfasis en transparencia de redes, con independencia de tecnología de hardware y que permite un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

RIA: Aplicaciones Enriquecidas de Internet (del inglés “Rich Internet Applications”) son un nuevo tipo de aplicaciones con más ventajas que las tradicionales aplicaciones Web. Esta surge como una combinación de las ventajas que ofrecen las aplicaciones Web y las aplicaciones tradicionales.

UML: El Lenguaje Unificado de Modelado (del inglés, “Unified Modeling Language”) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Bibliografía

Adobe System Incorporated, 2009. Adobe Adobe AIR. Disponible en: http://www.adobe.com/products/air/ [Accedido Febrero 11, 2009].

Adobe System Incorporated, 2008a. Adobe Labs - Adobe Flex Builder Linux Public Alpha. Disponible en: http://labs.adobe.com/technologies/flex/flexbuilder_linux/ [Accedido Febrero 11, 2009].

Adobe System Incorporated, 2008b. Adobe Open Source - Confluence. Disponible en: http://opensource.adobe.com/wiki/display/site/Home [Accedido Febrero 11, 2009].

Adobe Systems Inc., 2008a. Adobe Flash Player. *Adobe - Flash Player*. Disponible en: http://www.adobe.com/products/flash/about/ [Accedido Diciembre 8, 2008].

Adobe Systems Inc., 2008b. Adobe Flex. Disponible en: http://www.adobe.com/products/flex/ [Accedido Diciembre 8, 2008].

AJAX Animator Dev, 2008. Ajax Animator. Disponible en: http://antimatter15.110mb.com/phpfusion/news.php?readmore=29 [Accedido Febrero 4, 2009].

Cliff, J., 2008. Salasaga. Disponible en: http://www.salasaga.org/ [Accedido Febrero 4, 2009].

Daniel Fischer, 2008. swfmill swf2xml and xml2swf. *swfmill swf2xml and xml2swf*. Disponible en: http://swfmill.org/ [Accedido Junio 3, 2008].

Florian Delizy, 2007. UIRA » Unfreeze's. *Uira Unfreeze's*. Disponible en: http://www.unfreeze.net/?page_id=52 [Accedido Junio 3, 2008].

Grossman, Huang, 2006. ActionScript 3.0 overview. Disponible en: http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html [Accedido Febrero 9, 2009].

Kruchten, P., 2002. A Software Development Process for a Team of One.

Martín Olivera, Y. P. & Saez Villavicencio, A. C. Análisis de un IDE para múltiples plataformas con tecnologías y herramientas libres para desarrollar software educativo en formato multimedia.

Subsistema de gestión visual. Universidad de las Ciencias Informáticas, Ciudad de la Habana,Cuba, 2008.

Ming, 2008. “Libming Wiki”. Disponible en: http://www.libming.org/ [Accedido Junio 2, 2009].

Mika Palmu, 2008. FlashDevelop Open Source Flash. *FlashDevelop Open Source Flash*. Available at: http://osflash.org/flashdevelop [Accedido Junio 3, 2008].

Motion Tween, 2008. haXe - ¡Bienvenido a haXe! *haxe*. Available at: http://haxe.org/ [Accedido Junio 3, 2008].

Motion Tween, 2007. Motion-Twin. *Motion Tween*. Available at: http://www.mtasc.org/ [Accedido Junio 3, 2008].

Open Dialect Dev, 2008. Open Dialect – Trac. Available at: http://dialect.openmodeling.net/wiki [Accedido Febrero 4, 2009].

Sergi Perez Maña, 2007. Qflash - flash maker - website. *QFlash -flash maker - website*. Disponible en: http://qflash.sourceforge.net/webpage/ [Accedido Junio 3, 2008].

Shore and Warden, 2007. *The Art of Agile Development* Primera Edición., O’Reilly.

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/ [Accedido Junio 3, 2008].

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 [Accedido Junio 3, 2008].

Visual Paradigm, 2008. UML CASE Tools - Free for Learning UML, Cost-Effective for Business Solutions. Disponible en: http://www.visual-paradigm.com/product/vpuml/ [Accedido Febrero 11, 2009].

Scott W. Ambler, 2007. XP and the UML? Clearly the Wrong Question to be Asking Disponible en: http://www.agilemodeling.com/essays/xpUML.htm [Accedido Mayo 16, 2009].