Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Arquitectura para la creación de aplicaciones multimedia. MAPri

Autores:
Fabian Finalé Franqui
Yoennis Garrido Vargas

Tutores:
Ing. Michel Miranda Cairo
Ing. Abel Ernesto Lorente Rodríguez

Ciudad de la Habana, junio 2009
Año del 50 aniversario del triunfo de la Revolución

Para poner paz entre los hombres han de ser los adelantos de la ciencia.

José Martí

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ _____

Autor: Fabian Finalé Franqui Autor: Yoennis Garrido Vargas

_____ _____

Tutor: Michel Miranda Cairo Tutor: Abel Ernesto Lorente Rodríguez

DEDICATORIA

A mi abuela que no pudo verme graduado;
Viejuca, te extraño.

Fabian

A mi mamá, Arelis porque todo lo que soy es gracias a su amor y consejos que tanta falta me han hecho siempre.

A mi papá, Fidel por darme su apoyo en todo y aunque no lo demuestre mucho yo sé que me quiere.

A mi hermana, Dianelis por ser mi hermanita del alma y aunque tengamos nuestras diferencias yo la quiero mucho.

A mi hermano, Marcos, que aunque no hemos estado siempre juntos yo lo tengo siempre muy presente en mi corazón.

A mi novia, Yaima por ser la persona más linda de este mundo, y porque con su amor me ha hecho feliz estos cinco años.

A mis tíos, Aracelis y en especial a Alexis, que siempre me han brindado su amor y apoyo incondicional.

A mi abuela, Tila que la quiero un montón y ha sido mi segunda madre.

En fin a toda mi familia, porque este sueño es también parte de ellos.

Yoennis

AGRADECIMIENTOS

He llegado al final de este camino y quisiera expresar un profundo agradecimiento a quienes con su ayuda, apoyo y comprensión me alentaron a convertir este sueño en realidad.

A mis padres por lo que soy y por todo el tiempo que les robé pensando en mí, porque sin escatimar esfuerzo alguno han sacrificado gran parte de sus vidas para formarme y porque nunca podré pagar todos sus desvelos.

A mis hermanos Faidyt y Félix por confiar en mí y comprenderme. A los dos gracias por sobrellevar mis malcriadeces de hermano menor.

A Nidia, por estar a mi lado durante todo este tiempo y por todo el amor que me ha dado. Por comprenderme siempre y por no abandonarme nunca en los malos momentos. Gracias por ser quién eres.

A Libardito, por sacarme de apuros en más de una ocasión y por creer en mí.

A mis inseparables Eric, Humberto, Jorge y Jose, los sigo queriendo como el primer día.

A Yoennis, por toda su ayuda.

A Michel, Roberto y Yonny por todos los cabos que me tiraron.

A mis compañeros de grupo y a mis profesores por soportarme estos largos cinco años.

A todos ustedes, gracias.

Fabian

A mi mamá, que sin ti no hubiera podido realizar este sueño, porque tu amor, ayuda y dedicación han sido importante en toda mi vida de estudiante, por haberme dado la vida, ser mi ejemplo y porque este también es tu sueño.

A mi papá, por haber estado a mi lado, confiar en mí y apoyarme en todo.

A mis hermanos, que de una forma u otra siempre han estado conmigo y en especial a mi hermanita del alma.

A mi novia, por haber estado siempre a mi lado sin importar el momento o las circunstancias, por haber sido tan paciente, comprensiva y darme tanta felicidad.

A mis tíos, porque me han apoyado siempre y han confiado mucho en mi.

A mis abuelas, que me han brindado su amor y cariño.

A mis amigos, que de una forma u otra siempre han estado conmigo en los momentos buenos y malos, al piquete del Clan Moa, en especial: al Pikiri, al Enano, al Rafa, a Felipao, Keyler, Yoel, Ana, Yadira, Liutmila, Yoenny, con los que he compartidos grandes momentos de mi vida durante estos cinco años.

A mis colegas de la infancia, que siempre se han preocupado por mí, Zeudis y Reynaldo.

Al paquete Macromedia, Jorge, Jose, Fabian, Humberto y Erick, por haber compartido conmigo estos últimos cuatro años y brindarme su apoyo incondicional.

A mi colega de tesis Fabian, que sin su labor imprescindible no hubiese sido posible concretar este trabajo.

A mis compañeros de grupo, con quienes compartí estos inolvidables cinco años, por ser estupendas personas.

A mis compañeros de proyecto, Rolian, Humberto, Brian, Marlon, y otros, que han compartido conmigo en todo momento.

A mis profesores por inculcarme los conocimientos, a quieres admiro por su entrega y dedicación.

A Roberto, Yonny y Michel por toda la ayuda brindada.

Le agradezco a la Revolución Cubana por brindarme la posibilidad de estudiar en esta universidad de excelencia y en especial a nuestro eterno comandante en jefe Fidel Castro, por ser el forjador de este proyecto futuro.

A todos los que contribuyeron de una forma u otra a la realización de este trabajo, porque sin la ayuda de ustedes no hubiera podido realizar este sueño.

Yoennis

RESUMEN

Ante la necesidad de encaminar y unir, bajo líneas afines y reutilizables, el desarrollo de productos multimedia basados en la Plataforma Flash, surge la idea de crear un modelo de arquitectura que, respetando las normas tradicionales para la elaboración de Software Educativo, propiciara el desarrollo de las técnicas de trabajo y recortara el tiempo de realización de dichos productos.

En el presente trabajo de diploma: “Arquitectura para la creación de aplicaciones multimedia. MAPri” se persigue como objetivo principal desarrollar una arquitectura para ActionScript 3.0 basada en alternativas libres para la producción de aplicaciones multimedia en la UCI.

Para ello se realizó un análisis de la situación actual del desarrollo de productos multimedia en la Universidad de las Ciencias Informáticas, lo que posibilitó la realización de una arquitectura que permitiera la reutilización de código y componentes. Formando parte de esta solución se encuentran los Modelos de Desarrollo, constituyendo un ejemplo con todas las funcionalidades esenciales de las aplicaciones multimedia.

Palabras Claves

Micro Arquitectura, Framework, Plataforma Flash, ActionScript, Multimedia

ÍNDICE

INTRODUCCIÓN

Con el surgimiento y perfeccionamiento de las Tecnologías de la Información y las Comunicaciones (TIC), el mundo se ha visto inmerso en un proceso de cambios vertiginosos, encaminados al mejoramiento de la sociedad actual en la mayoría de sus esferas y la educación, uno de sus principales sectores, no ha estado exenta de toda esta revolución.

Actualmente, existen diversas tecnologías que contribuyen al mejoramiento del proceso de enseñanza-aprendizaje, fomentando el desarrollo educativo y garantizando la eficacia de los métodos pedagógicos utilizados. Dentro de esta diversidad tecnológica se destaca considerablemente el uso de las aplicaciones multimedia, permitiendo combinar de forma interactiva gran cantidad de información en forma de textos, videos, sonidos e imágenes.

La Universidad de las Ciencias Informáticas (UCI) es un centro educativo/productivo orientado a ser la mayor institución productora de software de Cuba, donde se han desarrollado un sinnúmero de aplicaciones incluyendo varias herramientas educativas, en su mayoría de tipo multimedia.

Debido al nivel de producción alcanzado en la UCI, han surgido diversas Comunidades de Desarrollo en las distintas ramas dentro del campo de las Ciencias Informáticas; ejemplo de ello es el nacimiento de la Comunidad de Desarrollo Primavera, cuyo principal objetivo consiste en darle vida, uniformidad, unidad y cooperación al Desarrollo Multimedia-Web (DMW) en la UCI. Primavera significó una nueva etapa, un nuevo aire y un nuevo estilo de trabajo colectivo orientado al desarrollo y la organización de los proyectos Multimedia-Web en la Universidad.

Sin embargo, a pesar de todo esto, no se tiene en cuenta ninguna arquitectura que permita encaminar, bajo líneas afines y reutilizables, la creación de aplicaciones multimedia basadas en formato ShockWave Flash (swf). Además, la mayoría de las herramientas utilizadas están sujetas a algún tipo de restricciones legales, pues casi la totalidad de las soluciones hoy existentes dependen, en mayor o menor medida, de herramientas privativas lo que trae consigo la comercialización de éstas aplicaciones a través de terceros países generando una considerable disminución en las ganancias de nuestro país.

Sumado a lo anterior, en la UCI se malgastan recursos monetarios y capital humano destinados a la producción de software con tecnología multimedia ya que no se utilizan una serie de pautas y estándares

que permitan la reutilización de códigos y componentes; retrasando el progreso de los proyectos y afectando directamente la calidad del producto final, lo que ocasiona un mayor costo de producción.

Después de analizar la situación existente el **problema científico** determinado es el siguiente: ¿Cómo desarrollar productos con tecnología multimedia basados en alternativas libres?

Se define como **objeto de estudio** el desarrollo de productos multimedia y software educativo en la UCI y como **campo de acción** el desarrollo de una arquitectura para ActionScript 3.0 basada en alternativas libres para la creación de aplicaciones multimedia en la UCI, siendo el **objetivo general** de la investigación desarrollar una arquitectura para ActionScript 3.0 basada en alternativas libres para la producción de aplicaciones multimedia en la UCI.

Para darle cumplimiento al presente trabajo se cuenta con varios **objetivos específicos** tales como:

- Integrar la arquitectura a los estándares de creación de aplicaciones multimedia.
- Implementar los modelos de desarrollo.
- Elaborar la documentación para las guías de desarrollo de la arquitectura.

Para encaminar la investigación con vista a resolver el problema planteado se propone la siguiente **idea a defender:** La creación de una arquitectura para ActionScript 3.0 basada en alternativas libres reduce considerablemente el tiempo de desarrollo de las aplicaciones multimedia y contribuye a la reutilización de código y componentes.

Para el desarrollo de la Micro-Arquitectura Primavera (MAPri) se cuenta con una serie de **tareas** que ayudarán a cumplimentar su construcción:

1. Realizar una revisión bibliográfica.
2. Fundamentar la investigación.
3. Seleccionar y fundamentar una metodología de desarrollo acorde a la magnitud del proyecto.
4. Investigar, seleccionar y documentar las herramientas y tecnologías a utilizar.
5. Evaluar los patrones de arquitectura y estándares de diseño a utilizar.

6. Realizar el diseño de la arquitectura.
7. Realizar la implementación por etapas de un prototipo funcional de MAPri.
8. Desarrollar por etapas un prototipo funcional del framework de MAPri.
9. Realizar la documentación de las clases que componen el prototipo funcional del framework que se entregue en cada iteración.
10. Implementar los modelos de desarrollo de MAPri en cada etapa.
11. Documentar las guías de desarrollo de la Micro-Arquitectura.
12. Realizar el documento de tesis.

FUNDAMENTACIÓN TEÓRICA

MICRO-ARQUITECTURA

PRIMAVERA

PRIMAVERA
MICRO-ARQUITECTURA
MAPri
FUNDAMENTACIÓN TEÓRICA

CAPÍTULO 1

Fundamentación Teórica

1.1. Introducción

Con la creciente demanda en la UCI de productos con tecnología multimedia soportadas en formato *.swf, ha crecido la necesidad de utilizar una arquitectura que permita la reutilización de código y componentes para disminuir el tiempo de desarrollo y los costos de producción de este tipo de software.

Estas arquitecturas, también llamadas frameworks, no son más que una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, pueden incluir soporte de programas, bibliotecas y un lenguaje que puede ser interpretado por otras aplicaciones para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Gracias a esto, el uso de los frameworks facilita la implementación de cualquier aplicación, sin hacer distinciones en las tecnologías que se utilicen, permitiendo así la reutilización de código y librerías y eliminando gran cantidad de errores de codificación.

Para el estudio de una solución que permita a la Universidad de las Ciencias Informáticas utilizar un framework para ActionScript 3.0 que se adapte a sus necesidades, este capítulo se centrará en la investigación de otras tecnologías y la existencia de soluciones similares basadas en alternativas libres que faciliten la implementación de aplicaciones multimedia.

1.2. Análisis de otras soluciones existentes

En el mercado de aplicaciones multimedia, constituye un estándar de facto el uso del ActionScript como lenguaje para su producción, sin embargo los desarrolladores siguen intentando realizar herramientas que

sean tanto novedosas como útiles y fáciles de usar. Algunos han creado gestores de contenidos, instaladores y diversos tipos de aplicaciones, otros se han dedicado más a la confección de un framework que facilite el trabajo en la creación de productos con tecnología multimedia. A continuación se describen los principales frameworks para ActionScript.

1.2.1. CASA lib

Es una librería flexible de ActionScript 2.0 y 3.0 diseñada para simplificar las tareas comunes y actuar como una base sólida y confiable para los proyectos. Proporciona un conjunto básico de clases, interfaces y utilidades con la finalidad de agilizar la implementación de líneas de código más fiables. Esta librería estandariza la carga de archivos externos con una interfaz de programación de aplicaciones (API por sus siglas en inglés) consistente y fácil de usar. Permite una administración de memoria mucho más fácil a partir de un método que elimina los elementos innecesarios para liberar recursos del ordenador. Incluye una documentación completa y meticulosa con ejemplos para cada clase y utilidad (Clinger, 2007).

Esta librería, a pesar de ser gratuita, está suscrita a restricciones legales (licencia BSD) lo que trae consigo que no pueda ser comercializado ningún producto que la utilice sin autorización de sus creadores.

1.2.2. JumpShip

Es un framework para ActionScript 3.0 construido con el patrón de diseño Modelo Vista Controlador (MVC) y orientado a tener en cuenta las mejores prácticas de MVC. Una de las cosas más singulares de JumpShip es la inclusión de un modelo de datos estándar que permite buscar, ordenar y enumerar automáticamente. Este modelo ofrece una estructura de datos increíblemente poderosa a través de todo el framework. Otra diferencia básica entre JumpShip y otros marcos MVC es su esfuerzo consciente por no basarse en el patrón Singleton como una manera fácil de crear referencias a otras clases (Zhang, 2008).

A pesar de las ventajas del framework, este se encuentra publicado en Google Code, un servidor de Google para hospedar proyectos, pero le deniega el acceso a Cuba por ser un país embargado.

1.2.3. ARP

Ariaware RIA Platform (ARP) es un framework basado en la arquitectura de patrones de diseño, la cual hace posible que equipos grandes de desarrolladores puedan trabajar eficientemente, estableciendo una delegación clara de responsabilidades y una comunicación a través de patrones similares (Balkan, 2004).

Este framework está orientado para la creación de Aplicaciones Ricas de Internet (RIAs por sus siglas en inglés) por lo que dificulta su utilización para la creación de aplicaciones multimedia.

1.2.4. SimpleAS3

Es un framework que, con la fortaleza del ActionScript 3.0, hace que el código fuente sea más parecido al ActionScript 2.0, simplificando el trabajo en cuanto a líneas de código. Facilita además la carga de textos, imágenes, documentos con formato *.xml y archivos *.swf (Tynjala, 2008).

A pesar de la simplicidad del código fuente en cuanto a la cantidad de líneas, este framework dificulta la implementación de una aplicación multimedia al existir escasa documentación, además, presenta errores a la hora de crear un proyecto nuevo.

1.2.5. Cairngorm

Es una microarquitectura ligera para las RIAs construidas en Adobe Flex o Adobe AIR. Implementando los más reconocidos patrones de diseño, Cairngorm ejemplifica y alienta las mejores prácticas para el desarrollo de RIAs adoptados por Adobe (The Cairngorm Team, 2008).

La mayor parte de este framework está diseñada para Adobe Flex, lo que dificulta su uso para la confección de aplicaciones multimedia.

1.2.6. Gaia

Es un framework de código abierto para ActionScript 2.0 y 3.0 diseñado para reducir drásticamente el tiempo de desarrollo. Gaia está destinado para cualquiera que haga sitios web en la plataforma flash y provee soluciones a los retos y tareas repetidas como la navegación, transiciones, pre-carga o administración de objetos. Brinda velocidad y flexibilidad en el flujo de trabajo y una API simple que proporciona acceso a las más poderosas características (Sacks, 2006).

Este framework está orientado a la realización de aplicaciones web por tanto se dificulta su uso para la creación de aplicaciones con tecnología multimedia.

1.2.7. LowRA

Low-level Rework on Actionscript (LowRA) es un framework para ActionScript 3.0 basado en el patrón Inversión de Control (IOC por sus siglas en inglés) soportado por un sistema de plug-ins y un poderoso modelo de objeto (Xavier, 2007).

Este framework carece de documentación por lo que se dificulta su uso.

1.2.8. VEGAS

Es un framework de código abierto basado en ECMAScript que puede ser utilizado tanto en ActionScript 2.0, 3.0 y ActionScript del lado del servidor. Provee muchas facilidades para ayudar a los desarrolladores a crear RIAs (Stoica, 2007).

Este framework está diseñado para desarrollar aplicaciones web, por lo que es imposible utilizarlo en la realización de una aplicación multimedia.

1.3. Descripción general del objeto de estudio

Actualmente la gran mayoría de las aplicaciones con tecnología multimedia y de software educativo que se desarrollan en la Universidad de las Ciencias Informáticas están basadas en el formato *.swf. La casi totalidad de las herramientas utilizadas para desarrollar aplicaciones de este tipo están, en mayor o menor medida, sujetas a algún tipo de restricciones legales, dentro de estas herramientas la más usada es Adobe Flash Professional 8. Este software, perteneciente a Adobe Systems Incorporated, es quien ha controlado al formato *.swf casi desde sus inicios, y desde entonces ha venido perfeccionándose desde las funcionalidades más básicas hasta brindar las facilidades que hoy ofrece, incluyendo un editor de ActionScript, un ambiente amigable para el diseño y la animación de la interfaz, un conjunto de componentes prediseñados así como streaming bidireccional de audio y video. A pesar de todas estas facilidades ofrecidas por el Adobe Flash, es legalmente imposible su uso en la UCI pues la licencia de este producto prohíbe su venta a Cuba.

Por otra parte, en la UCI no se utiliza ningún estándar o arquitectura que permita la reutilización de código y componentes, por lo que el tiempo de desarrollo de las aplicaciones multimedia se prolonga más de lo necesario, trayendo consigo un aumento de los costos de producción y un mayor uso de capital humano.

1.4. Identificación de la audiencia

La arquitectura a implementar irá dirigida a todos los desarrolladores de aplicaciones con tecnología multimedia en la Universidad. Además, puede ser extendido tanto a todo el país como a nivel internacional, ya que pretende facilitar la creación de productos multimedia, permitiendo su confección en apenas horas o días, en dependencia de la magnitud del trabajo y de la experiencia que se tenga con la arquitectura. Para el uso de la misma es necesario tener conocimientos básicos sobre ActionScript y Programación Orientada a Objetos (POO).

1.5. Tendencias y tecnologías

Ante el incesante avance de las tecnologías, la sociedad, ávida de nuevas herramientas y funcionalidades, exige a los desarrolladores de software nuevos retos y nuevas concepciones para satisfacer sus exigencias, cada vez más ambiciosas. Para satisfacer estas exigencias, los desarrolladores deben buscar nuevas ideas surgiendo así nuevas metodologías y formas de desarrollo que permitan confeccionar productos cada vez más complejos.

Como parte de estas nuevas ideas surgen las aplicaciones multimedia, abriendo así las puertas a una tecnología totalmente revolucionaria, una tecnología que, a pesar de basarse en un concepto tan antiguo como el hombre, se encuentra en constante desarrollo y constituye un gigantesco paso de avance para la informática.

Se suele identificar como multimedia a la integración de dos o más medios de comunicación que pueden ser controlados o manipulados por el usuario en una computadora. O sea, es un sistema informático interactivo, controlable por el usuario, que integra diferentes medios como el texto, el vídeo, la imagen, el sonido y las animaciones (Colectivo de Autores del INSTED, 2006), pero mientras no se defina adecuadamente a la “Multimedia”, se le seguirá comprendiendo como lo nuevo dentro de la técnica y la capacidad intelectual de combinar los medios de comunicación y el Hombre- Máquina (Alejos, 2007).

El desarrollo de aplicaciones multimedia ha sido testigo de la creación de varios lenguajes y paradigmas de programación. En la actualidad, se sigue la tendencia de usar la POO por permitir abstraer problemas de la vida real y ActionScript por ser fácil, flexible, robusto y orientado a objetos.

Igualmente, los formatos más utilizados por aplicaciones multimedia son *.flv para videos, *.jpeg y *.png para imágenes y *.mp3 para sonidos.

***.flv:** Flash Video es un formato de archivo propietario usado para transmitir video sobre internet usando el reproductor Adobe Flash Player y pueden ser incrustados dentro de archivos *.swf. Puede ser visto en la mayoría de los sistemas operativos. El audio en los archivos *.flv se encuentra generalmente codificado en *.mp3.

***.jpeg:** Joint Photographic Experts Group es un método comúnmente utilizado para la compresión de imágenes fotográficas, permitiendo ajustar el grado de reducción y seleccionando así el compromiso existente entre el tamaño de almacenamiento y la calidad de la imagen. A pesar de ser un estándar de compresión, es a menudo considerado como un formato de archivo y alcanza normalmente una compresión de 10 a 1 con pocas pérdidas perceptibles en la calidad de la imagen. Lleva el nombre de la comisión que creó la norma, integrada desde sus inicios por la fusión de varias agrupaciones en un intento de compartir y desarrollar su experiencia en la digitalización de imágenes.

***.png:** Portable Network Graphics es un formato gráfico basado en un algoritmo de compresión sin pérdidas para mapas de bits (bitmaps) no sujeto a patentes. Fue desarrollado en buena parte para solventar las deficiencias del formato *.gif y permite almacenar imágenes con una mayor profundidad de contraste y otros importantes datos.

***.mp3:** MPEG-1 Audio Layer 3 es un formato de audio digital comprimido con pérdida desarrollado por el Moving Picture Experts Group (MPEG) para formar parte de la versión 1 del formato de video *.mpeg. El mp3 estándar es de 44Hz y posee un flujo de bits (bitrate) de 128 kbps por la relación de calidad/tamaño.

1.6. Metodologías y Herramientas Case a utilizar

Las metodologías de desarrollo de software engloban procedimientos, técnicas, documentación y herramientas que tienen como base fundamental el fortalecimiento de las actividades de los proyectos y procesos de desarrollo de un software. Actualmente no existe una metodología universal que le haga

frente con éxito a cualquier proyecto de desarrollo de software, ni que se adapte a todo tipo de sistema, debido a que las características y recursos de cada equipo de desarrollo no siempre son las mismas. Existen propuestas más tradicionales que establecen rigurosamente las actividades involucradas, los artefactos que se deben producir, y las notaciones y herramientas que se usarán; y otras que se caracterizan por estar centradas en el personal que compone el equipo de trabajo y la reducción al máximo del número de artefactos producidos. Estas últimas son conocidas como Metodologías Ágiles.

1.6.1. Metodologías Tradicionales

1.6.1.1. Proceso Unificado de Desarrollo (RUP)

El Proceso Unificado de Desarrollo (RUP) es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo a través de un Lenguaje Unificado de Modelado (UML), constituyendo la metodología estándar más utilizada para el análisis, desarrollo, implementación y documentación de software orientado a objetos.

RUP cuenta con tres características fundamentales, dirigido por los Casos de Uso, iterativo incremental y centrado en la arquitectura. Define cuatro fases esenciales (Inicio, Elaboración, Construcción y Transición) y nueve flujos de trabajos; seis de Ingeniería (Modelado del Negocio, Requerimientos, Análisis y Diseño, Implementación, Prueba y Despliegue) y tres de apoyo (Gestión de la Configuración, Gestión de Proyecto y Ambiente) (Letelier, 2006).

Esta metodología incluye artefactos (productos tangibles del proceso) y roles (papel que desempeña una persona en un determinado momento a lo largo del proceso), y se basa en cinco principios claves (adaptar el proceso, balancear prioridades, demostrar valor iterativamente, elevar el nivel de abstracción y enfocarse en la calidad).

Características de RUP

- Establece una forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo).
- Pretende implementar las mejores prácticas en Ingeniería de Software.

- Establece un desarrollo iterativo.
- Permite la administración de requisitos.
- Hace uso de una arquitectura basada en componentes.
- Permite llevar un control de cambios.
- Permite realizar un modelado visual del software.
- Verifica la calidad del software.

A pesar de las ventajas que posee esta metodología, requiere de una evaluación de riesgos compleja, presenta una excesiva flexibilidad para algunos proyectos y el cliente debe ser capaz de describir y entender a un gran nivel de detalle para que los desarrolladores puedan acordar un alcance del proyecto con él. No es flexible para el desarrollo de proyectos pequeños que pueden estar sujetos a constantes cambios en sus requerimientos; debido a que un pequeño cambio en uno de ellos, provocaría que se tenga que generar nuevamente toda la documentación y prolongaría el tiempo de desarrollo del proyecto. Por estos motivos se hace un poco engorroso el uso de RUP para guiar el desarrollo de la arquitectura que se quiere desarrollar.

1.6.2. Metodologías Ágiles

En una reunión celebrada en febrero de 2001 en Utah, EEUU, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. Varias de las denominadas metodologías ágiles ya estaban siendo utilizadas con éxito en proyectos reales, pero les faltaba una mayor difusión y reconocimiento.

Tras esta reunión se creó *The Agile Alliance*, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía "ágil" (Letelier, 2006).

1.6.2.1. Crystal

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener definidas las políticas de trabajo en equipo. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros). Implica poca disciplina durante el proceso de desarrollo. Confía en la autorregulación y libera el proceso de codificación. Las revisiones se realizan al final de la iteración. Los métodos Crystal no prescriben las prácticas de desarrollo, las herramientas o los productos que pueden usarse, pudiendo combinarse con otros métodos como SCRUM y XP (Mony, 2008).

Los roles definidos por Crystal son:

- Patrocinador Ejecutivo (Executive Sponsor)
- Jefe de Proyecto (Project Manager)
- Experto en el Dominio (Domain Expert)
- Experto de uso (Usage Expert)
- Programador Diseñador (Designer-Programmer)
- Diseñador UI (Designer)
- Realizador de Pruebas (Tester)

- Programador Técnico (Technical)

Debido a la poca bibliografía que existe sobre dicha metodología se hace difícil su uso en el desarrollo de la arquitectura.

1.6.2.2. SCRUM

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle, define un marco para la gestión de proyectos que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para lograr coordinación e integración (Penadés, 2006).

Scrum define métodos de gestión y control para complementar la aplicación de otros métodos ágiles como XP que, centrados en prácticas de tipo técnico, carecen de ellas.

<u>Principio de SCRUM</u>

- Los equipos son auto-gestionados.
- Una vez dimensionadas las tareas no es posible agregarles trabajo extra.
- Se realizan reuniones diarias en las que los miembros del equipo se plantean 3 cuestiones:
    - ✓ ¿Qué has hecho desde la última revisión?
    - ✓ ¿Qué obstáculos te impiden cumplir la meta?
    - ✓ ¿Qué vas a hacer antes de la próxima reunión?
- Las iteraciones de desarrollo tienen una frecuencia inferior a un mes, al final de las cuales se presenta el resultado a los externos del equipo de desarrollo, y se realiza una planificación de la siguiente iteración, guiada por el cliente (Jorquera, 2008).

SCRUM no es apto para todo tipo de proyecto, debido a que pequeños cambios en los requisitos de un software durante una iteración no son corregidos hasta terminar dicha iteración y comenzar la nueva. Además no consta con una amplia documentación, provocando que su uso sea un poco engorroso a la hora de aplicarse a la arquitectura.

1.6.2.3. Extreme Programming (XP)

XP es una metodología creada por Kent Beck, que se centra en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo.

Se basa en la realimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios. Se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre.

Características

- Su desarrollo es iterativo e incremental con pequeñas mejoras, unas tras otras.
- Se realizan pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Se establece la programación por parejas, recomendándose que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto, para lograr una mayor calidad del código fuente escrito.
- Hay una frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Se corrigen de todos los errores antes de añadir una nueva funcionalidad y se hacen entregas frecuentes.

- Hay una refactorización del código, es decir, se reescriben ciertas partes del código para aumentar su legibilidad y mantenimiento pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.

- La propiedad del código compartida, ya que en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores sean detectados.

- Exige simplicidad en el código, siendo esta la mejor manera de que las cosas funcionen. Cuando todo funcione se podrán añadir funcionalidades si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo (Álvarez, y otros, 2007).

Roles XP

Aunque en otras fuentes de información aparecen algunas variaciones y extensiones de roles XP, en este apartado se describen los roles de acuerdo con la propuesta original de Beck.

- Programador: El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.

- Cliente: El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.

- Encargado de pruebas (*Tester*): El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.

- Encargado de seguimiento (*Tracker*): El encargado de seguimiento proporciona la realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones.

- Entrenador (*Coach*): Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

- Consultor: Es un miembro externo al equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.

- Gestor (*Big boss*): Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

El ciclo de vida ideal de XP consiste de seis fases (Exploración, Planificación de la Entrega (*Release*), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto) (Letelier, y otros, 2006).

XP no es un modelo de procesos ni un marco de trabajo, sino un conjunto de 24 prácticas que se complementan unas a otras y que ofrecen una base sólida para un óptimo desempeño, alta productividad e inestimables beneficios (Pruebas, Diseño simple, Entregas pequeñas, Refactorización, Programación en parejas, Cliente in-situ) y deben implementarse en un entorno de desarrollo cuya cultura se base en cinco valores:

1. Simplicidad: Enfocado más en un diseño sencillo del código generando sólo la documentación indispensable.
2. Comunicación: Potenciada por el desarrollo en pares y la presencia del cliente, además de la simplicidad en cuanto al código.
3. Retroalimentación: Propiciada por el protagonismo del cliente que participa activamente y por el trabajo en ciclos cortos.
4. Coraje: Enfrentando decisiones en ocasiones complejas que pudieran afectar el tiempo de desarrollo y la calidad del producto.
5. Respeto: Basado en estimar en toda su magnitud el trabajo de los demás (Jorquera, 2008).

Las prácticas de XP se encuentran agrupadas en cuatros principales grupos que abarcan todo lo referente con el desarrollo del sistema:

- Requisito de Análisis y Planificación (Historias de Usuarios), ciclo semanal, ciclo trimestral, despliegue incremental, participación real del cliente, ámbito de aplicación del contrato negociado y pago por uso).
- Equipo y Factores Humanos (trabajo en equipo, constante información del trabajo realizado, buena energía de trabajo, programar en dúos, mantener una estrecha relación con el equipo de continuidad y reducir los equipos de desarrollo).
- Diseño (diseño incremental, realizar pruebas y analizar detalladamente las causas, para evitar cometer el mismo error de nuevo).
- Codificación y Liberación del Software (integración continua, realizar el código y hacer las pruebas, compartir el código para todo el equipo de programadores, existencia de una única base de código y hacer un despliegue diario de todo lo que se ha realizado en el día).

Ventajas

- Es apropiado para entornos volátiles.
- Se está preparado para el cambio, significando una reducción en su coste.
- La planificación es más transparente para los clientes ya que conocen las fechas de entrega de las funcionalidades vitales para su negocio.
- Permite definir en cada iteración cuáles son los objetivos de la siguiente.
- Permite tener una retroalimentación por parte de los usuarios.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

Después de todo el estudio previamente realizado se puede llegar a la conclusión de que históricamente, las metodologías tradicionales han intentado abordar la mayor cantidad de situaciones de contexto principalmente a proyectos de gran magnitud, ya que exigen un gran esfuerzo para ser adaptadas, sobre todo en proyectos pequeños y con requisitos muy cambiantes, como la arquitectura a desarrollar. Sin

embargo las metodologías ágiles ofrecen una solución casi a la medida para una gran cantidad de proyectos con características similares a las de la arquitectura que se desea implementar, sobre todo XP (Extreme Programming) que, aunque para algunos resulta muy arriesgada su utilización por una serie de inconvenientes y restricciones, es la que mejor se ajusta a las necesidades y condiciones de un equipo de trabajo compuesto por sólo dos personas y que cuenta con un corto período de tiempo para el desarrollo de una solución final. Todo esto es posible gracias a la sencillez que presenta XP, tanto en su aprendizaje como en su aplicación, reduciendo los costos de implantación en un equipo de desarrollo.

1.7. Herramientas CASE

Las herramientas CASE (Ingeniería de Software Asistida por Computadoras) son aplicaciones informáticas que tienen como objetivo fundamental solucionar y afrontar los problemas de una mala calidad de software y una documentación inadecuada. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo de un determinado proyecto o software, ya que brindan la posibilidad de realizar cálculos de costos, generan código fuente automáticamente de un diseño previamente dado, poseen compilación automática, ayudan con la documentación y juegan un papel importante en la detección de errores.

Por tales ventajas algunas personas han dado su criterio personal sobre el concepto CASE:

En *Terminology for Software Engineering and Computer-aided Software Engineering* por *B.Terry & D.Logee, Software Engineering Notes* en Abril de 1990, CASE es definido como:

“Herramientas individuales para ayudar al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software o mantenimiento del mismo” (Universidad Politécnica de Valencia, 2004).

En *The CASE Experience, Carma McClure, BYTE* de Abril 1989 *p.235* se ofrece la siguiente definición:

“Una combinación de herramientas de software y metodologías de desarrollo” (Universidad Politécnica de Valencia, 2004).

Además, estas herramientas son ajustables a distintas metodologías de desarrollo de software, mejoran la productividad del desarrollo y mantenimiento del software, aumentan la calidad de los mismos y ayudan a la reutilización del software, portabilidad y estandarización de la documentación.

1.7.1. Rational Rose Enterprise Edition Suite 2003

Rational Rose Enterprise Edition es una de las herramientas más poderosas del modelado visual para el análisis y diseño de sistemas basados en objetos. Se utiliza para modelar un sistema antes de proceder a construirlo. Cubre todo el ciclo de vida de un proyecto: concepción y formalización del modelo, construcción de los componentes, transición a los usuarios y la certificación de las distintas fases de desarrollo del software.

Rational Rose propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

Características

- Desarrollo Iterativo: Utiliza un proceso de desarrollo iterativo controlado, donde el desarrollo se lleva a cabo en una secuencia de iteraciones.

- Ingeniería Inversa: Proporciona mecanismos para realizar la denominada Ingeniería Inversa, a partir del código de un programa, se puede obtener su diseño.

- Trabajo en Grupo: Permite varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

- Generador de Código: Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML (Moreno Martínez, 2000).

Esta herramienta CASE no genera código para ActionScript, por lo que no cumple con las necesidades de la arquitectura.

1.7.2. Visual Paradigm 6.04

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones con una mejor calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Se encuentra soportada en todos los sistemas operativos y aunque precisa de licencia, la universidad tiene acceso a la misma.

Características

- Interoperabilidad con modelos UML2 (metamodelos UML 2.x para plataforma Eclipse).
- Ingeniería inversa - Código a modelo, código a diagrama.
- Ingeniería inversa Java, C++, Esquemas XML, XML, .NET exe/dll, CORBA IDL.
- Generación de código - Modelo a código, diagrama a código.
- Diagramas de flujo de datos.
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos - Desde Sistemas Gestores de Bases de Datos (DBMS) existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.

- Importación y exportación de ficheros XMLI.
- Integración con Microsoft Visio - Dibujo de diagramas UML con plantillas de MS Visio (Visual Paradigm International, 2007).

Visual Paradigm for UML 6.04 se escoge como herramienta CASE, para llevar una buena organización y planificación de los diagramas y modelados del framework a desarrollar. Además, es el único de su tipo que genera código para ActionScript, lo cual es de vital importancia para el desarrollo de la arquitectura, porque ahorra tiempo y líneas de código a implementar.

1.8. UML. El Lenguaje de Modelado Unificado

UML (Lenguaje Unificado de Modelado) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software, y entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir.

UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar.

Características del UML

- UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores. No tiene propietario y está basado en el común acuerdo de gran parte de la comunidad informática.
- UML no pretende ser un método de desarrollo completo. No incluye un proceso de desarrollo paso a paso. UML incluye todos los conceptos que se consideran necesarios para utilizar un proceso moderno iterativo, basado en construir una sólida arquitectura para resolver requisitos dirigidos por casos de uso.
- Debe ser tan simple como sea posible pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesita construir. UML necesita ser lo suficientemente expresivo para manejar todos

los conceptos que se originan en un sistema moderno, tales como la concurrencia y distribución, así como también los mecanismos de la ingeniería de software, como son la encapsulación y componentes.

- Debe ser un lenguaje universal, como cualquier lenguaje de propósito general.
- Pretende imponer un estándar mundial.
- Es orientado a objetos, permitiéndole al programador que organice su programa de acuerdo con abstracciones de más alto nivel, siendo estas más cercanas a la forma de pensar de las personas (Mariño2007).

1.9. Lenguaje de programación

ActionScript: es el lenguaje de POO integrado en plataformas Flash que posibilita el desarrollo de complejas aplicaciones multimedia y sitios web dinámicos bajo esta tecnología. Su campo de aplicación crece cada día, abarcando aplicaciones tan diversas como el desarrollo de juegos, simulaciones, presentaciones interactivas y animación dinámica con sorprendentes efectos visuales. Fue reconocido por primera vez en el mundo con el lanzamiento de la versión 4 de Flash, y desde entonces hasta la fecha, ha ido ampliándose poco a poco, llegando a niveles de dinamismo y versatilidad muy altos en la versión 9 (Adobe Flash CS3) de Flash.

1.9.1. ¿Por qué ActionScript 3.0 como lenguaje de programación?

ActionScript 3.0 (AS3) es un potente lenguaje de POO que marca un paso importante en la evolución de las capacidades en tiempo de ejecución de Flash Player. La motivación de conducción de ActionScript 3.0 es crear un lenguaje muy adecuado para la rápida construcción de RIAs, las que se han convertido en una parte esencial de la experiencia web. Está basado en ECMAScript, especificación de estándar de codificación publicada por ECMA International actualmente aceptada como el estándar ISO 16262.

ActionScript 3.0 aumenta las posibilidades de creación de *scripts* de las versiones anteriores. Se ha diseñado para facilitar la creación de aplicaciones muy complejas con conjuntos de datos voluminosos y bases de código reutilizables, fundamentalmente orientadas a objetos. No requiere necesariamente de

Adobe Flash Player 9 para la ejecución de su contenido. Una de las desventajas que corrige AS3 con respecto a las versiones anteriores, es el tratamiento de eventos, ya contiene en su framework un paquete de clases exclusivamente dedicadas a su gestión.

ActionScript 3.0 introduce una nueva y altamente optimizada máquina virtual (AVM2), que supera de manera espectacular el rendimiento posible de AVM1 utilizada por las versiones ActionScript 1.0 y 2.0, ejecutando el código AS3 hasta 10 veces más rápido.

ActionScript 3.0 ofrece una completa aplicación de ECMAScript para XML (E4X), recientemente normalizadas como ECMA-357. E4X ofrece un fluido juego de construcciones de lenguaje para manipular archivos *.xml, rompiendo con las tradicionales APIs de análisis existentes en versiones anteriores. E4X agiliza el desarrollo de aplicaciones que manipulan archivos *.xml mediante una reducción drástica de la cantidad de código necesario (Grossman, y otros, 2006).

Debido a esto y a las grandes ventajas que brinda AS3 con respecto a la programación orientada a objetos, se decide desarrollar la arquitectura en dicho lenguaje de programación.

1.10. Herramientas a utilizar

1.10.1. FlashDevelop 3.0.0 RC2

Es un editor de código con soporte para ActionScript 2.0, ActionScript 3.0, MXML, y HaXe. Es libre y de código abierto. Es fácil de integrar con Adobe Flash y con compiladores de líneas de comandos. Tiene integrado un Administrador de Proyectos y posee completamiento de código para XML, MXML y HTML. Posibilita la navegación de códigos e incluye generadores de documentación para ActionScript. Está liberado bajo la licencia MIT (FlashDevelop Development Team, 2009).

1.10.2. TortoiseSVN 1.4.5

Es un cliente Subversion, implementado como una extensión al shell de Windows. Es software libre liberado bajo la licencia GNU GPL. Puede ser usado sin un entorno de desarrollo y está disponible en 28 idiomas diferentes. Decora los íconos con pequeñas imágenes mostrando qué archivos o directorios necesitan ser enviados al repositorio.

1.10.3. OpenOffice.org 3.0.1

Es una suite ofimática de software libre y código abierto de distribución gratuita que incluye herramientas como procesador de textos, hoja de cálculo, presentaciones, herramientas para el dibujo vectorial y base de datos. Está disponible para muchas plataformas como Microsoft Windows y sistemas de tipo Unix con el Sistema X Window como GNU/Linux, BSD, Solaris y Mac OS X. También está disponible para Mac OS X un programa similar derivado denominado NeoOffice. OpenOffice está pensado para ser altamente compatible con Microsoft Office, con quien compite. Soporta el estándar ISO OpenDocument con lo que es fácil el intercambio de documentos con muchos otros programas, y puede ser utilizado sin costo alguno.

1.10.4. Mozilla Firefox 3.0.6

Es un navegador de internet libre y de código abierto descendiente de Mozilla Application Suite desarrollado por la Corporación Mozilla, la Fundación Mozilla y un gran número de voluntarios externos. Para visualizar páginas web, Firefox usa el motor de renderizado Gecko, que implementa algunos estándares web actuales además de otras funciones, algunas de las cuales están destinadas a anticipar probables adiciones a los estándares web. Incluye navegación por pestañas, corrector ortográfico, búsqueda progresiva, marcadores dinámicos, un administrador de descargas y un sistema de búsqueda integrado que utiliza el motor de búsqueda que desee el usuario. Además se pueden añadir funciones a través de complementos desarrolladas por terceros. Una de estas funciones es el manejador de referencias Zotero.

1.10.5. Zotero 1.0.9

Es una extensión libre para el navegador Firefox, que permite a los usuarios recolectar, administrar y citar investigaciones de todo tipo de orígenes del navegador. Es parcialmente una aplicación de administración de referencias, usada para administrar bibliografías y referencias al escribir ensayos y artículos. Se integra además con Microsoft Office y con OpenOffice.org.

1.11. Conclusiones

En el presente capítulo se realizó una investigación sobre el estado del arte de los distintos frameworks existentes para la confección de aplicaciones con tecnología multimedia así como el estudio de las metodologías de desarrollo y herramientas CASE. Una vez concluido, se pudo observar que ninguna de las soluciones existentes se ajusta a las necesidades, ya que la mayoría están suscritas a algún tipo de restricciones legales o son orientadas a la web. Después de haber analizado las metodologías existentes se decidió que por sus facilidades, documentación, y flexibilidad se utilizaría XP y además para la realización de la implementación del framework de la arquitectura se acordó utilizar FlashDevelop 3.0.0 RC2 por ser una herramienta liberada bajo licencia MIT y brindar una serie de funcionalidades para el trabajo con ActionScript 3.0. Como herramienta CASE se seleccionó Visual Paradigm 6.04 apoyado en UML. Una vez definidos estos aspectos, es posible pasar al siguiente capítulo donde se presenta la solución propuesta.

CARACTERÍSTICAS DEL SISTEMA
EXPLORACIÓN Y PLANIFICACIÓN

MICRO-ARQUITECTURA

PRIMAVERA

PRIMAVERA
MICRO-ARQUITECTURA

MAPi

CARACTERÍSTICAS DEL SISTEMA
EXPLORACIÓN Y PLANIFICACIÓN

CAPÍTULO 2

Características del Sistema. Exploración y Planificación

2.1. Introducción

El presente capítulo tiene como objetivo hacer una valoración de las principales características del sistema a desarrollar. Se detallan las necesidades de los usuarios, describiéndose las funcionalidades que serán objeto de automatización. Se hace alusión a las fases de exploración y planificación, propias de la metodología de desarrollo utilizada para la implementación del sistema que se propone y se exponen los artefactos generados durante el transcurso de las mismas.

2.2. Descripción de los procesos vinculados al campo de acción

Actualmente, en la Universidad de las Ciencias Informáticas en el proceso de desarrollo de productos multimedia, no se aplican estándares o una arquitectura que permita la reutilización de código y componentes, trayendo consigo que el tiempo de desarrollo de las aplicaciones multimedia se prolongue más de lo necesario, provocando un aumento en los costos de producción y un mayor uso del capital humano.

2.2.1. Flujo actual del proceso

El desarrollo de este proceso se realiza de la siguiente manera: Una vez que se firma el contrato del producto multimedia que se desea obtener o se decide desarrollar una aplicación para beneficio propio de la Universidad; la UCI escoge el equipo de trabajo que se encargará de la creación del software y, seguidamente, se seleccionan las herramientas a utilizar para desarrollarlo. Una vez obtenidos los requerimientos y necesidades por parte del cliente, el equipo de desarrollo se encarga de darle

respuestas, ya sean los especialistas de diseño, analistas de sistema, arquitectos, jefes de proyectos o programadores. Estos últimos son los encargados de implementar las funcionalidades que debe cumplir el software; para la realización de esta tarea no cuentan con ninguna arquitectura o framework, que les permita reutilizar código y componentes, por lo que es necesario reelaborar el código desde cero para cada nueva aplicación.

2.3. Propuesta del sistema

El presente trabajo propone la implementación de un sistema basado en la utilización conjunta de varias herramientas libres (Adobe Flex Open Source SDK como compilador y Flash Develop como editor de código). MAPri propone la creación de paquetes de clases en ActionScript 3.0 que revolucionarán el desarrollo de productos multimedia en la Universidad de Ciencias Informáticas, en la que son llevados a cabo una gran cantidad de proyectos de esta índole. Para lograr este objetivo, cuenta con un estándar de desarrollo de software bajo modelos, documentación y guías, que representan un factor fundamental en la uniformidad y conexión entre los métodos de trabajos colectivos e individuales de cada estructura de la misma.

MAPri está concebida en tres partes:

El Framework: es la base de las aplicaciones MAPri. Compuesto por un gran número de clases gestoras de información, es la herramienta de desarrollo de los patrones/modelos de la arquitectura, representando la estructura vertebral de la misma.

Cada clase está concebida con un objetivo específico y afín con todas las demás, dependiendo de su aplicación a cada modelo, y su uso es crucial debido a la importancia que le imprime a los entornos multimedia el trabajo con POO y la organización, reusabilidad y experiencia que esto significa.

Basado en su importancia y uso, las clases del framework han sido clasificadas en las siguientes categorías:

- Navegación: gestiona el entorno de navegación dentro del producto multimedia. Pueden ser definidas desde el propio IDE o desde documentos XML (IDE/XML).

- Interactividad: útiles para la creación de juegos muy comunes en los productos multimedia, ya sean cuestionarios, juegos de arrastre, marcado de respuestas, crucigramas, entre otros. Todo esto vinculado a documentos XML.

- Estilos de Información: encargados de la representación visual del contenido y de lograr un formato uniforme respecto al entorno y a los objetos/textos del producto. Personalizable desde hojas de estilo (CSS) para los textos.

- Gestión de Recursos: representa la base del entorno y propicia la información completa de las rutas de las imágenes, textos, videos, sonidos y animaciones a incorporar en la aplicación multimedia. Resulta muy útil a la hora de la creación de colecciones multimedia y entornos similares, ya que facilita la modificación de las rutas de los recursos.

- Componentes: acompañados de clases, aunque no las tienen como principal objetivo. Son compontes prefabricados a partir del trabajo en clases como símbolos, botones, menús, ventanas, etc.

La documentación de MAPri ofrece la posibilidad de obtener información precisa de los métodos de las clases, especificaciones de éstas y ejemplos concretos de código en formato *.swf. Presenta la base de uso del framework y los detalles de trabajo de sus clases de forma colectiva e independiente. Al igual que el framework de clases, está en constante desarrollo y evolución en función de su uso y objetivos concretos.

Los modelos de desarrollo son la base de modelado y estructuración de los proyectos desarrollados bajo MAPri. Están formados por dos partes fundamentales:

- Guías de Desarrollo: compuestas por las vías de producción de las aplicaciones multimedia con la utilización del framework y elaboradas a partir de la experiencia de cada producto “nuevo” respecto a la estructura creada en uno anterior, y se basa en la documentación, bajo ciertos parámetros técnicos que trazan un camino de desarrollo de la aplicación. Como tal es la experiencia de cada proyecto a partir de la elaboración de un documento en el cual se expliquen y tracen los “pasos” a seguir para llevar a cabo la creación de productos similares. Estos son agregados a MAPri, junto con todas las clases independientes/nuevas utilizadas por los desarrolladores del mismo.

- Modelos de desarrollo: son el ejemplo factible de las Guías de Desarrollo y, como tal, modelos “vacíos” en cuanto a información, listos para ser entregados en días a las manos de un cliente, luego de ser modificados y editados por los diseñadores, programadores y guionistas, es decir, son aplicaciones multimedia sin contenido, una especie de maqueta funcional.

2.3.1. Personal relacionado con el sistema

Se define como persona relacionada con el sistema a aquella que interactúa con el framework y obtiene un resultado de su interacción, además de todas aquellas personas que están de una forma u otra vinculadas a la realización de la arquitectura como tal, ya que intervienen en ella, sin obtener un resultado.

Tabla 2.1 Personas relacionadas con el sistema

Personas relacionadas con el sistema.	Justificación
Desarrollador	Es la persona encargada de utilizar la arquitectura y apoyarse en ella para la creación de productos multimedia, por lo que es necesario que tenga conocimientos sobre ActionScript y Programación Orientada a Objetos.
Jefe de Proyecto	Es la persona encargada de asesorar y dar seguimiento del estado del proceso de desarrollo.

2.3.2. Lista de Reserva

Después de conocido el personal relacionado y los objetivos que se quieren lograr, se puede pasar al análisis de las funcionalidades que debe cumplir el framework para darle respuesta a los mismos. Para ello se enumerarán mediante una Lista de Reserva, las funcionalidades que el sistema debe ser capaz de cumplir. Dentro de ella se encuentran incluidas las acciones que pueden ser ejecutadas por los desarrolladores que interactúen con el framework y las condiciones externas a determinar por el mismo.

De acuerdo a lo antes mencionado el sistema debe ser capaz de:

1. Crear actividad interactiva Sopa de Letras.

2. Crear actividad interactiva Crucigramas.
3. Crear actividad interactiva Pareo.
4. Crear actividad interactiva Completamiento de Oraciones.
5. Crear actividad interactiva Selección Simple.
6. Validar respuestas de actividades interactivas.
7. Configurar la aplicación.
8. Navegar de forma lineal.
9. Navegar en cascada.
10. Reproducir fondo musical.
11. Pausar fondo musical.
12. Cargar imágenes.
13. Cargar video.
14. Cargar texto.
15. Cargar sonido.
16. Crear reproductor.
17. Reproducir película.
18. Pausar película.
19. Bajar volumen de la película.
20. Seleccionar una posición para reproducir película.

21. Crear temporizador.

22. Imprimir documento.

23. Buscar contenidos.

24. Validar búsqueda.

2.4. Fase de Exploración

La metodología de desarrollo XP comienza con su fase de exploración, durante esta etapa los clientes plantean a grandes rasgos las HU que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. Las estimaciones realizadas en dicha fase son primarias (ya que estarán basadas en datos de muy alto nivel), y podrían variar cuando se analicen más detalladamente en cada iteración. Esta fase dura típicamente un par de semanas, y el resultado es una visión general del sistema, y un plazo total estimado.

2.4.1. Historias de usuarios

Las HU son utilizadas en la metodología de desarrollo ágil XP para la especificación de requerimientos de un sistema (acompañadas de pruebas de aceptación). Estas son una forma rápida de administrar las necesidades de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos, debido a que un requerimiento de software es descrito de forma concreta y sencilla utilizando el lenguaje común del usuario. Las HU permiten responder rápidamente a los requerimientos cambiantes y aunque se redactan desde las perspectivas de los clientes, también los desarrolladores pueden brindar ayuda en la identificación de las mismas.

La diferencia más importante entre estas historias y los tradicionales documentos de especificación funcional se encuentra en el nivel de detalle requerido. Las historias de usuario deben tener el detalle mínimo como para que los programadores puedan realizar una estimación poco riesgosa del tiempo que llevará su desarrollo. Deben poder ser programadas en un tiempo entre una y tres semanas. Si la

estimación es superior a tres semanas, debe ser dividida en dos o más historias. Si es menos de una semana, se debe combinar con otra historia de usuario.

Plantilla de Historias de Usuarios:

Historia de Usuario	
No.: *Número sucesivo a partir de 1.*	**Nombre:** *Identifica la historia de usuario en cuestión.*
Usuario: *Quien ejecuta la historia de usuario.*	
Prioridad en el Negocio: *Define la relevancia e impacto de la historia de usuario para el negocio de acuerdo a las necesidades del usuario.*	**Nivel de Complejidad:** *Define la dificultad técnica que supone desarrollar la historia de usuario desde el punto de vista del programador.*
Puntos de Estimación: *Permiten estimar duración de implementación.*	**Iteración Asignada:** *Precisa la iteración a la que pertenece la historia de usuario.*
Descripción: *Explica en qué consiste la historia de usuario, teniendo en cuenta las acciones realizadas por el usuario y la respuesta brindada por el sistema.*	
Información adicional (Observaciones): *Información extra que se estime agregar para hacer más comprensible la historia de usuario. Por ejemplo: conceptos, post-condiciones, relación con otros requisitos, etc.*	

Como resultado del trabajo realizado durante las fases de exploración se identificaron las siguientes historias de usuario:

Tabla 2.4.1. HU Actividad Interactiva.

Historia de Usuario	
No.: 1	**Nombre:** Actividad Interactiva
Usuario: Usuario Común.	
Prioridad en el Negocio: Media	**Nivel de Complejidad:** Alta

Puntos de Estimación: 2	**Iteración Asignada:** 1
Descripción: Se le debe brindar la posibilidad, a cualquier persona interesada en medir sus habilidades y conocimientos adquiridos mediante el estudio de los contenidos del producto multimedia, que lo puedan hacer de forma amena a través de juegos didácticos y ejercicios interactivos que contribuyan a un mejor aprendizaje de los contenidos. En cada una de las distintas tipologías de ejercicio propuestas, el sistema debe informar al usuario si dicha actividad ha sido realizada satisfactoriamente, una vez termine el ejercicio.	
Información adicional: En algunas tipologías, como por ejemplo: Crucigramas, Completamiento de Oraciones y Selección Simple), el sistema debe brindarle la posibilidad al usuario de volver a intentar el ejercicio, en caso de que se haya equivocado anteriormente. Se hace referencia a las funcionalidades 1, 2, 3, 4, 5 y 6, enumeradas en el listado de reservas.	

Tabla 2.4.2. HU Carga de Configuración.

Historia de Usuario	
No.: 2	**Nombre:** Carga de Configuración
Usuario: Desarrollador.	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 1
Descripción: El desarrollador debe configurar la aplicación especificando el tipo de navegación, el tipo de fondo musical, los tipos de medias que contiene así como su cantidad y las rutas de cada cual.	
Información adicional: Esta configuración se debe hacer mediante la modificación del XML de configuración. Se hace referencia a la funcionalidad número 7, enunciada en la lista de reservas.	

Tabla 2.4.3. HU Navegación.

Historia de Usuario	
No.: 3	**Nombre:** Navegación
Usuario: Usuario Común.	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Medio
Puntos de Estimación: 1	**Iteración Asignada:** 1
Descripción: Se le brinda la posibilidad al usuario de navegar por la aplicación de acorde al tipo de navegación requerida, ya sea de forma lineal o en cascada. Estos tipos de navegación son los más frecuentes en los productos con tecnologías multimedia.	
Información adicional: Se hace referencia a las funcionalidades 8 y 9, enumeradas en el listado de reservas.	

Tabla 2.4.4. HU Fondo Musical.

Historia de Usuario	
No.: 4	**Nombre:** Fondo Musical
Usuario: Usuario Común.	
Prioridad en el Negocio: Baja	**Nivel de Complejidad:** Bajo
Puntos de Estimación: 1	**Iteración Asignada:** 1
Descripción: Se le brinda la posibilidad al usuario de pausar o reproducir el fondo musical de la aplicación.	
Información adicional: El framework debe permitir que la reproducción de los sonidos de fondo pueda ser de forma lineal o aleatoria. Se hace referencia a las funcionalidades 10 y 11, enumeradas en el listado de reservas.	

Tabla 2.4.5. HU Carga de Medias.

Historia de Usuario	
No.: 5	**Nombre:** Carga de Medias
Usuario: Usuario Común.	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Bajo
Puntos de Estimación: 1	**Iteración Asignada:** 1
Descripción: A medida que el usuario navegue por la aplicación, se efectúa la carga de los distintos tipos de medias existentes.	
Información adicional: Estas medias pueden ser textos, imágenes, videos o sonidos, y sus datos deben ser especificados por los desarrolladores en sus correspondientes archivos *.xml. Se hace referencia a las funcionalidades 12, 13, 14 y 15, enumeradas en el listado de reservas.	

Tabla 2.4.6. HU Reproductor.

Historia de Usuario	
No.: 6	**Nombre:** Reproductor
Usuario: Usuario Común.	
Prioridad en el Negocio: Media	**Nivel de Complejidad:** Medio
Puntos de Estimación: 2	**Iteración Asignada:** 2
Descripción: En caso de que la aplicación tenga un reproductor, este debe permitir realizar funciones básicas como detener, pausar o comenzar la reproducción, subir y bajar el volumen de la misma así como brindar la posibilidad de que el usuario vaya a un punto específico del archivo mediante una barra de desplazamiento.	
Información adicional: El reproductor debe contar además con un temporizador que indique la longitud del archivo y la posición actual del cabezal del reproductor. Se hace referencia a las funcionalidades 16, 17, 18, 19, 20 y 21, enumeradas en el listado de reservas.	

Tabla 2.4.7. HU Galería de Imágenes.

Historia de Usuario	
No.: 7	**Nombre:** Galería de Imágenes
Usuario: Usuario Común.	
Prioridad en el Negocio: Alta	**Nivel de Complejidad:** Alta
Puntos de Estimación: 1	**Iteración Asignada:** 2
Descripción: Se le debe brindar la posibilidad de acceder a cualquier persona interesada en ver las imágenes relacionadas con el producto multimedia, mediante una galería de imágenes.	
Información adicional: El sistema debe brindar la posibilidad de que las imágenes se puedan ampliar a medida que los usuarios acceden a ellas, con el objetivo de lograr una mejor apreciación de las mismas. Se hace referencia a la funcionalidad número 12, enunciada en la lista de reservas.	

Tabla 2.4.8. HU Impresión de Archivos.

Historia de Usuario	
No.: 8	**Nombre:** Impresión de Archivos
Usuario: Usuario Común.	
Prioridad en el Negocio: Media	**Nivel de Complejidad:** Medio
Puntos de Estimación: 2	**Iteración Asignada:** 3
Descripción: La aplicación debe permitir que el usuario imprima un documento específico.	
Información adicional: El usuario debe especificar los datos de la impresión. Se hace referencia a la funcionalidad número 22, enunciada en la lista de reservas.	

Tabla 2.4.9. HU Buscador.

Historia de Usuario	
No.: 9	**Nombre:** Buscador
Usuario: Usuario Común.	
Prioridad en el Negocio: Media	**Nivel de Complejidad:** Alta
Puntos de Estimación: 2	**Iteración Asignada:** 3
Descripción: Se le debe brindar la posibilidad, a cualquier persona interesada en buscar información referente algún tema que sea de su interés, en todo el contenido tratado en el producto multimedia.	
Información adicional: El sistema debe mostrar a los usuarios el resultado obtenido de la búsqueda realizada y validar que el criterio de búsqueda sea mayor de tres letras. Se hace referencia a las funcionalidades 23 y 24, enumeradas en el listado de reservas.	

2.5. Fase de planificación

En esta fase el cliente establece la prioridad de cada historia de usuario, y seguidamente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Las estimaciones de esfuerzo asociado a la implementación de las HU las establecen los programadores utilizando como medida el punto de estimación. Un punto de estimación se considera como una semana ideal de trabajo, donde los miembros de los equipos de desarrollo trabajan el tiempo planeado sin ningún tipo de interrupción, este punto de estimación que se utiliza para representar la semana ideal, es de 5 días. Por otra parte, en esta fase, el equipo de desarrollo mantiene un registro de la “velocidad” de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos de estimación correspondientes a las historias de usuario que fueron terminadas en la última iteración. Esta fase dura solamente unos pocos días.

2.5.1. Estimación de esfuerzos por historia de usuario

Para el buen desarrollo del sistema propuesto, se realizó una estimación para cada una de las historias de usuario identificadas, llegando a los resultados que se muestran a continuación:

Tabla 2.5.1. Estimación de esfuerzo por historia de usuario

Historias de usuario	Puntos de estimación
Actividad Interactiva	2 semanas
Carga de Configuración	1 semana
Navegación	1 semana
Fondo Musical	1 semana
Carga de Medias	1 semana
Reproductor	2 semanas
Galería de Imágenes	1 semana
Impresión de Archivos	2 semanas
Buscador	2 semanas

2.5.2. Plan de iteraciones

Una vez descritas las HU y estimado el esfuerzo propuesto para la realización de cada una de ellas, se procede a realizar la planificación de la etapa de implementación del sistema. En este plan se establece cuántas iteraciones serán necesarias realizar sobre el sistema para su terminación. El plan de iteraciones puede incluir indicaciones sobre cuáles HU se incluirán en un *release*, lo cual debería ser consistente con el contenido de una o dos iteraciones.

En relación con lo antes tratado se decide realizar el sistema en 3 iteraciones, las cuales se describen detalladamente a continuación:

Iteración 1

Esta iteración tiene como objetivo darle cumplimiento a las HU que se consideraron de mayor importancia para el desarrollo del framework. Al concluir dicha iteración se contará con todas las funcionalidades descritas en las HU 1, 2, 3, 4 y 5, las cuales hacen alusión al tipo de navegación que pueden tener los

modelos de desarrollo (ya sea lineal o en cascada), al módulo de actividades interactivas, a todo lo referente a la carga de la configuración, tipos de fondos musicales y todo lo involucrado con la carga de medias. Además se tendrá la primera versión de prueba, que contará con dos modelos de desarrollo que incorporan todas las funcionalidades antes vistas, éstos modelos se le presentarán al cliente con el objetivo de obtener una retroalimentación del mismo para posteriores iteraciones del producto.

Iteración 2

Esta iteración tiene como finalidad desarrollar las HU que responden a los números 6 y 7. Dichas HU son las que brindan las funcionalidades de reproducción de contenidos y visualización de imágenes a través de una galería de imágenes. Además permiten a los usuarios controlar el estado de la reproducción de los contenidos, mediante algunas funciones básicas como detener, pausar o comenzar la reproducción, subir y bajar el volumen de la misma así como brindar la posibilidad de que el usuario vaya a un punto específico del archivo mediante una barra de desplazamiento. La versión que se obtendrá de esta iteración en unión con la entregada en la iteración anterior se le dará al cliente para verificar si cumple con las necesidades antes acordadas con él.

Iteración 3

Esta es la última iteración del framework, en la que se dará cumplimiento a las HU 8 y 9. Dichas HU cumplen con las funcionalidades de impresión de documentos contenidos en el producto multimedia y brindan la posibilidad de hacer búsquedas de información referente a algún tema que sea de interés para los usuarios que interactúen con el software. Estas HU son integradas con el resultado de las iteraciones anteriores y como fruto de esta integración se obtendrá la versión 1.0 del producto final, que contiene tres modelos de desarrollos, que muestran la utilidad del framework y la arquitectura desarrollada. A partir de este momento el software será puesto a un proceso de prueba para evaluar el desempeño del mismo.

2.5.3. Plan de duración de las iteraciones

Como parte del ciclo de vida de un proyecto guiado por la metodología de desarrollo de software XP, se crea el plan de duración de cada una de las iteraciones que se llevarán a cabo durante el desarrollo del proyecto. Este plan tiene como finalidad mostrar la duración de cada iteración, así como el orden en que serán implementadas las HU en cada una de las mismas.

Tabla 2.5.3. Plan de duración de las iteraciones

Iteración	Historias de Usuario	Duración total iteraciones
Iteración 1	Actividad Interactiva	6 semanas La entrega se realizará en la primera semana de marzo.
	Carga de Configuración	
	Navegación	
	Fondo Musical	
	Carga de Medias	
Iteración 2	Reproductor	3 semanas La entrega se realizará en la 2da semana de abril.
	Galería de Imágenes	
Iteración 3	Buscador	4 semanas La entrega se realizará en la 3ra semana de mayo.
	Impresión de Archivos	

2.6. Conclusiones

Con la realización del presente capítulo se comenzó a desarrollar la propuesta de solución que se desea implementar, se analizó el flujo de proceso que se lleva a cabo actualmente en la UCI, y se definieron las posibles mejoras de automatización de estos procesos, con el propósito de lograr acelerar, mejorar y guiar el desarrollo del mismo. Para esto se definieron los requerimientos funcionales del framework propuesto y se identificaron los requerimientos no funcionales que se deben tener en cuenta en la construcción del mismo. Además se trata todo lo referente a las dos primeras fases de la metodología de desarrollo de software a utilizar, fase de exploración y planificación del sistema, donde se documentaron todos los artefactos generados en el transcurso de las mismas. Y quedó plasmado que el desarrollo del framework se realizará en 3 iteraciones y programado en pareja, como lo propone la metodología utilizada.

MICRO-ARQUITECTURA

PRIMAVERA

PRIMAVERA
MICRO-ARQUITECTURA

MAPFRE

CONSTRUCCIÓN DE LA SOLUCIÓN PROPUESTA

CAPÍTULO 3

Construcción de la Solución Propuesta

3.1. Introducción

En este capítulo se construye la solución propuesta de forma iterativa, tal y como indica la metodología XP. Se presenta el diagrama de clases del framework de la arquitectura, así como el estándar de codificación utilizado para la implementación de las clases de MAPri.

3.2. Diseño de la Solución Propuesta

La metodología de desarrollo XP establece prácticas especializadas, que inciden directamente en la realización y elaboración del diseño de un software, sin embargo no requiere que la representación del sistema sea mediante diagramas de clases basados en UML, sino que pueden emplearse indistintamente sencillos esquemas descritos en pizarras u otras técnicas como las tarjetas CRC (Contenido, Responsabilidad y Colaboración). No obstante el empleo de los diagramas UML pueden ser utilizados siempre y cuando contribuyan en el mejoramiento de la comunicación del equipo de desarrollo, no sean muy extensos y no requieran de mucho tiempo para su creación.

3.2.1. Tarjetas CRC

Las tarjetas CRC forman parte de las técnicas propuestas por algunos de los creadores de la metodología ágil XP (Ward Cunningham y Kent Beck), con el objetivo de obtener un diseño simple y que no incurra en

la implementación de funcionalidades que no son necesarias. Esta técnica de modelado permite entender las características del sistema pensando en términos de objetos y clases.

Debido a la gran facilidad de uso y entendimiento, que propician dichas tarjetas, el equipo de desarrollo del presente trabajo, decidió utilizarlas para diseñar el framework que se desea desarrollar. Además de crear un pequeño diagrama de clases con el objetivo de que se tenga una mejor visión del diseño del framework, para aquellos desarrolladores que en un futuro pudieran trabajar en él, y no tenga mucha experiencia en el uso de las tarjetas CRC. (Ver Anexo 1).

Tabla 3.2.1 Plantilla para las Tarjetas CRC

Tarjeta CRC	
Clase: *Nombre de la clase que se está modelando.*	
Súper Clase: *Nombre de la clase padre en la herencia.*	
Sub Clase(s): *Nombre de la(s) clase(s) hija en la herencia.*	
Responsabilidades: *Es una descripción de alto nivel del propósito de la clase.*	**Colaboraciones:** *Indica con cuáles otras clases se requiere relación para cumplir la responsabilidad.*

Tabla 3.2.2 Tarjeta CRC MAPri

Tarjeta CRC	
Clase: MAPri	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades: Clase principal del framework y es la encargada de instanciar la clase que configura todos los elementos de la aplicación multimedia.	**Colaboraciones:** Clase Configuración

Tabla 3.2.3 Tarjeta CRC Configuración

Tarjeta CRC	
Clase: Configuración	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades: Clase encargada de configurar todos los elementos de la multimedia. Crea una instancia de la clase Carga_de_XML, carga el *.xml de configuración y crea instancias de las clases necesarias según se indique en este *.xml.	**Colaboraciones:** Clase Carga_de_XML Clase Ruta Clase Navegación Clase Navegación_Lineal Clase Navegación_en_Cascada Clase Fondo_Musical Clase Fondo_Musical_ Lineal Clase Fondo_Musical_ Aleatorio Clase Paginado

Tabla 3.2.4 Tarjeta CRC Ruta

Tarjeta CRC	
Clase: Ruta	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades: Clase gestora de las rutas de la aplicación multimedia. Permite hacer Búsquedas de Rutas y ser modificadas inclusive en tiempo de ejecución.	**Colaboraciones:**

Tabla 3.2.5 Tarjeta CRC Navegación

Tarjeta CRC	
Clase: MAPri	
Súper Clase: -	
Sub Clase(s): Navegación Lineal, Navegación_en_Cascada	
Responsabilidades: Clase abstracta encargada de la navegación. Establece los elementos comunes a los distintos tipos de navegación.	**Colaboraciones:**

Tabla 3.2.6 Tarjeta CRC Navegación_Lineal

Tarjeta CRC	
Clase: Navegación_Lineal	
Súper Clase: Navegación	
Sub Clase(s): -	
Responsabilidades: Clase encargada de la navegación lineal, fotograma a fotograma.	**Colaboraciones:** Clase Carga_de_XML Clase Carga_de_TextoXML Clase Paginado

Tabla 3.2.7 Tarjeta CRC Navegación_en_Cascada

Tarjeta CRC	
Clase: Navegación_en_Cascada	
Súper Clase: Navegación	
Sub Clase(s): -	
Responsabilidades: Clase encargada de la navegación en cascada, es decir, de manera no secuencial.	**Colaboraciones:** Clase Carga_de_XML Clase Carga_de_TextoXML Clase Paginado

Tabla 3.2.8 Tarjeta CRC Paginado

Tarjeta CRC	
Clase: Paginado	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades: Clase encargada de crear las páginas de la aplicación multimedia. Esta clase crea las páginas insertando las medias especificadas en el *.xml que contiene el paginado.	**Colaboraciones:** Clase Carga_de_Imagen Clase Carga_de_XML Clase Carga_de_TextoXML Clase Scroll

Tabla 3.2.9 Tarjeta CRC Carga_de_Medias

Tarjeta CRC	
Clase: Carga_de_Medias	
Súper Clase: -	
Sub Clase(s): Carga_de_XML, Carga_de_Imagen, Fondo_Musical, Carga_de_Estilo, Juego	
Responsabilidades: Clase abstracta encargada de la carga de medias. Establece los elementos comunes a los distintos tipos de carga de medias.	**Colaboraciones:**

Tabla 3.2.10 Tarjeta CRC Carga_de_XML

Tarjeta CRC	
Clase: Carga_de_XML	
Súper Clase: Carga_de_Medias	
Sub Clase(s): Carga_de_TextoXML	
Responsabilidades: Clase encargada de la carga y del parseo de los archivos *.xml.	**Colaboraciones:**

Tabla 3.2.11 Tarjeta CRC Carga_de_TextoXML

Tarjeta CRC	
Clase: Carga_de_TextoXML	
Súper Clase: Carga_de_XML	
Sub Clase(s): -	
Responsabilidades: Clase encargada de cargar los textos de la multimedia en formato XML y asignárselos a un campo de texto.	**Colaboraciones:** Clase Carga_de_Estilo

Tabla 3.2.12 Tarjeta CRC Carga_de_Imagen

Tarjeta CRC	
Clase: Carga_de_Imagen	
Súper Clase: Carga_de_Medias	
Sub Clase(s): -	
Responsabilidades: Clase encargada de la carga dinámica de las imágenes.	**Colaboraciones:**

Tabla 3.2.13 Tarjeta CRC Fondo_Musical

Tarjeta CRC	
Clase: Fondo_Musical	
Súper Clase: Carga_de_Medias	
Sub Clase(s): Fondo_Musical_Lineal, Fondo_Musical_Aleatorio	
Responsabilidades: Clase abstracta encargada de gestionar la reproducción de la música de fondo de la aplicación multimedia.	**Colaboraciones:**

Tabla 3.2.14 Tarjeta CRC Fondo_Musical_Lineal

Tarjeta CRC	
Clase: Fondo_Musical_Lineal	
Súper Clase: Fondo_Musical	
Sub Clase(s): -	
Responsabilidades: Clase encargada de gestionar la reproducción de forma lineal y cíclica de la música de fondo de la aplicación multimedia.	**Colaboraciones:**

Tabla 3.2.15 Tarjeta CRC Fondo_Musical_Aleatorio

Tarjeta CRC	
Clase: Fondo_Musical_Aleatorio	
Súper Clase: Fondo_Musical	
Sub Clase(s): -	
Responsabilidades: Clase encargada de gestionar la reproducción de forma aleatoria y cíclica de la música de fondo de la aplicación multimedia.	**Colaboraciones:**

Tabla 3.2.16 Tarjeta CRC Fondo_Musical_Aleatorio

Tarjeta CRC	
Clase: Fondo_Musical_Aleatorio	
Súper Clase: Fondo_Musical	
Sub Clase(s): -	
Responsabilidades: Clase encargada de gestionar la reproducción de forma aleatoria y cíclica de la música de fondo de la aplicación multimedia.	**Colaboraciones:**

Tabla 3.2.17 Tarjeta CRC Scroll

Tarjeta CRC	
Clase: Scroll	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades: Clase encargada de desplazar el texto en caso de que no pueda mostrarse todo a la vez.	**Colaboraciones:**

Tabla 3.2.18 Tarjeta CRC Juego

Tarjeta CRC	
Clase: Juego	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades: Clase encargada de crear instancias de cada una de las actividades interactivas, con el objetivo de crear el juego didáctico que desee el cliente.	**Colaboraciones:** Clase Sopa_de_Plabras Clase Crucigrama Clase Selección_Simple Clase Completar_Oraciones Clase Memoria

Tabla 3.2.19 Tarjeta CRC Juego_de_Caracteres

Tarjeta CRC	
Clase: Juego_de_Caracteres	
Súper Clase: MovieClip	
Sub Clase(s): Sopa_de_Palabras, Crucigrama	
Responsabilidades: Maneja todo lo referente a la información de cada una de las actividades interactivas que requieren de una matriz, para la representación de los datos que conforman el juego y se encarga de cargar los datos desde el XML externo que contiene la información de la actividad.	**Colaboraciones:** Clase Palabra. Clase Carga_de_XML

Tabla 3.2.20 Tarjeta CRC Palabra

Tarjeta CRC	
Clase: Palabra	
Súper Clase: -	
Sub Clase(s): -	
Responsabilidades: Esta clase maneja todos los datos referentes a las palabras utilizadas en las actividades interactivas Sopa de Palabras, Crucigrama.	**Colaboraciones:**

Tabla 3.2.21 Tarjeta CRC Sopa_de_Palabra

Tarjeta CRC	
Clase: Sopa_de_Palabras	
Súper Clase: Juego_de_Caracteres	
Sub Clase(s): -	
Responsabilidades: Crear la actividad interactiva Sopa de Palabras con todas las funcionalidades descritas por el cliente respecto a dicha actividad.	**Colaboraciones:** Clase Juego_de_Caracteres

Tabla 3.2.22. Tarjeta CRC Crucigrama

Tarjeta CRC	
Clase: Crucigrama	
Súper Clase: Juego_de_Caracteres	
Sub Clase(s): -	
Responsabilidades: Crear la actividad interactiva Crucigrama con todas las funcionalidades descritas por el cliente respecto a dicha actividad.	**Colaboraciones:** Clase Juego_de_Caracteres

Tabla 3.2.23 Tarjeta CRC Selección_Simple

Tarjeta CRC	
Clase: Selección_Simple	
Súper Clase: MovieClip	
Sub Clase(s): -	
Responsabilidades: Crear la actividad interactiva Selección Simple con todas las funcionalidades descritas por el cliente respecto a dicha actividad.	**Colaboraciones:** Clase Carga_de_XML

Tabla 3.2.24 Tarjeta CRC Completar_Oraciones

Tarjeta CRC	
Clase: Completar_Oraciones	
Súper Clase: MovieClip	
Sub Clase(s): -	
Responsabilidades: Crear la actividad interactiva Completar Oraciones con todas las funcionalidades descritas por el cliente respecto a dicha actividad.	**Colaboraciones:** Clase Carga_de_XML

Tabla 3.2.25. Tarjeta CRC Memoria

Tarjeta CRC	
Clase: Memoria	
Súper Clase: MovieClip	
Sub Clase(s): -	
Responsabilidades: Crear la actividad interactiva Memoria con todas las funcionalidades descritas por el cliente respecto a dicha actividad.	**Colaboraciones:** Clase Carga_de_XML

3.2.2. Arquitectura del sistema

La Arquitectura del Software aporta una visión abstracta de alto nivel, posponiendo el detalle de cada uno de los módulos definidos a pasos posteriores al diseño. La definición oficial del término Arquitectura del Software, fue pronunciada por la IEEE estándar 1471-2000: “La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución”. El objetivo principal de la Arquitectura del Software es aportar elementos que ayuden a la toma de decisiones y, al mismo tiempo, proporcionar conceptos y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto. Para conseguirlo, la Arquitectura del Software construye abstracciones, materializándolas en forma de diagramas comentados.

Para el desarrollo del sistema propuesto se tomó en cuenta el estilo de programación por capas, cuyo objetivo primordial es la separación de la lógica de negocios de la lógica de diseño; un ejemplo básico de esto consiste en separar la capa de datos de la capa de presentación al usuario, que en este caso sería separar la lógica de programación del framework que propone la arquitectura, de los modelos de desarrollos que se le presentan al cliente.

Ventajas de esta Arquitectura

- El desarrollo se puede llevar a cabo en varios niveles.
- Desarrollos paralelos (en cada capa).
- Aplicaciones más robustas debido al encapsulamiento.
- En caso de que sobrevenga algún cambio, sólo se ataca al nivel requerido sin tener que revisar entre código mezclado.
- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica).
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nueva funcionalidad).
- Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware.

3.2.1.1. Diagrama de Componentes

Los diagramas de componentes representan cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Estos diagramas prevalecen en el campo de la arquitectura de software, pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema.

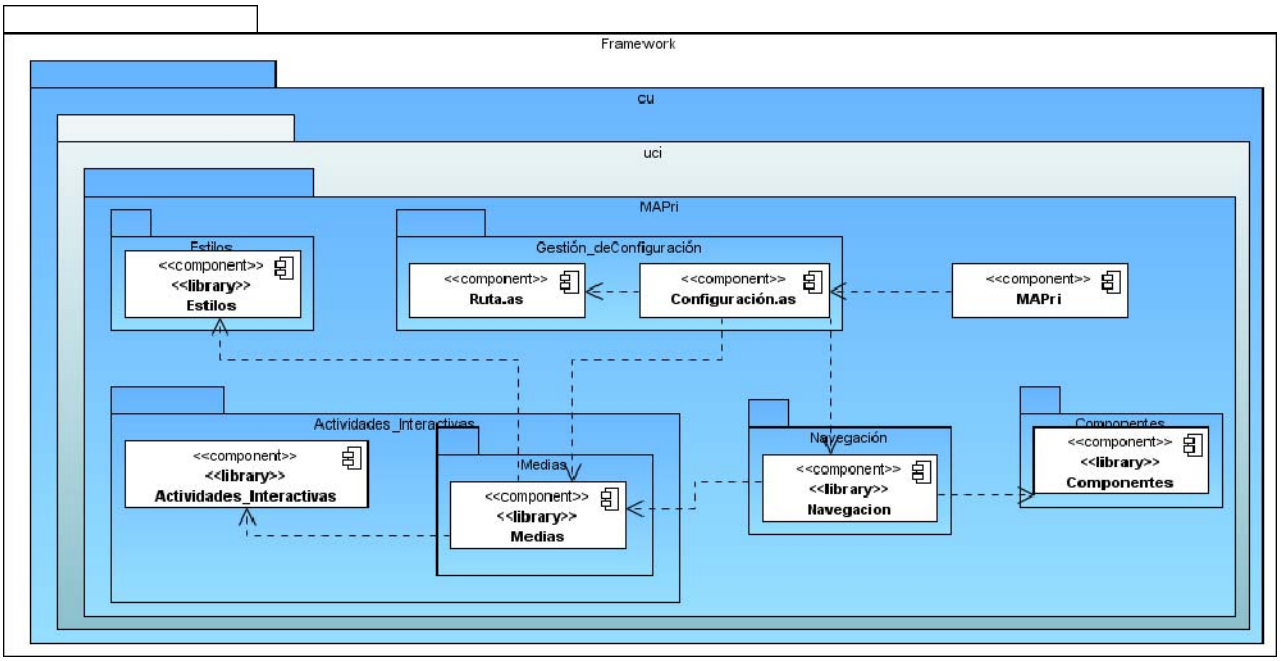

Figura 3.1 Diagrama de Componentes del Framework de MAPri.

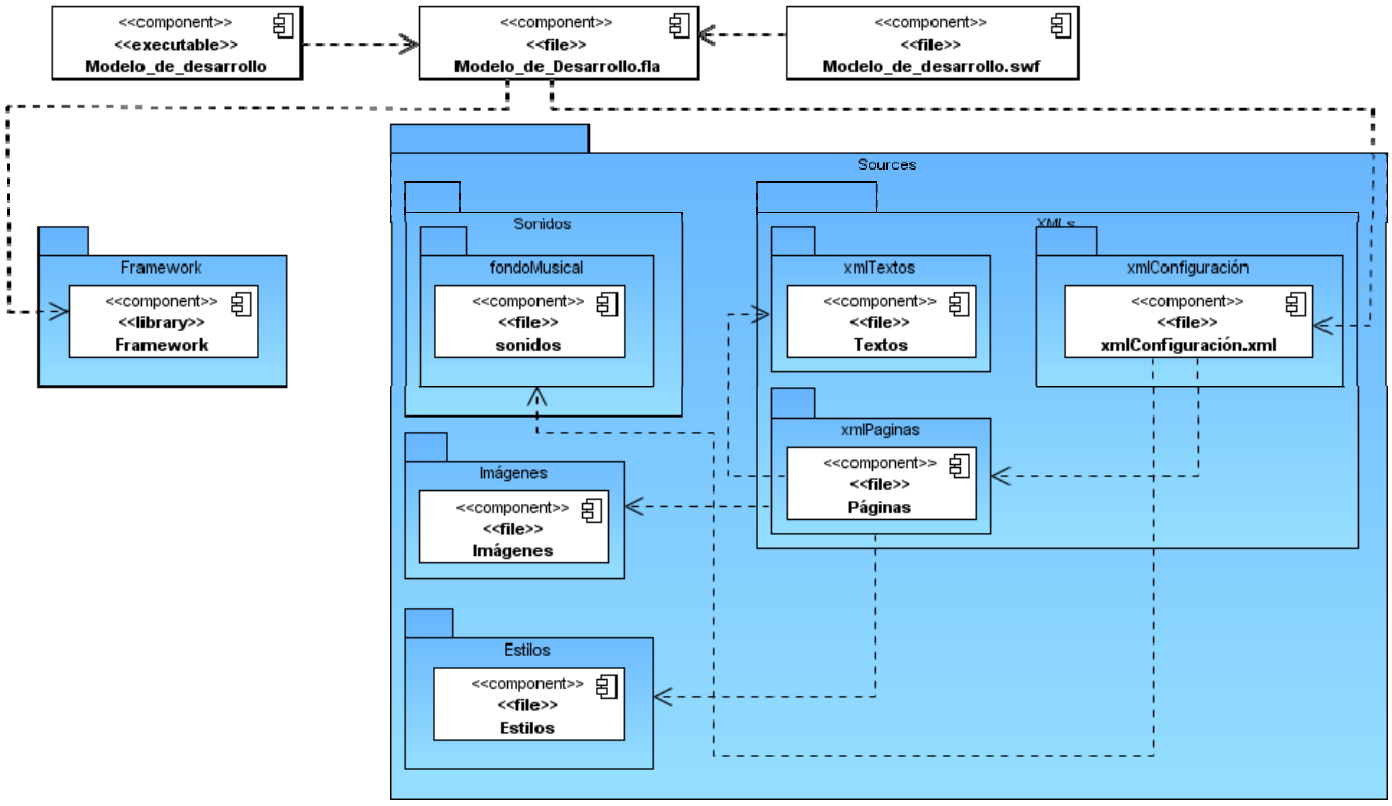

Figura 3.2 Diagrama de Componentes de los Modelos de Desarrollo de MAPri.

3.2.3. Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y tienen como objetivo proporcionar elementos reutilizables en el diseño de sistemas, así como evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.

Para la implementación del framework de la arquitectura se utilizaron los siguientes patrones:

Patrón Experto: ¿Cuál es un principio general para asignar responsabilidades a los objetos?

Asignar una responsabilidad al experto en información (la clase que tiene la información necesaria para la realización de la asignación). Un ejemplo del uso de este patrón en el framework de MAPri es la clase Navegación.

Patrón Creador: ¿Quién debería ser el responsable de la creación de una nueva instancia de alguna clase?

Asignar a la clase B la responsabilidad de crear una instancia de clase A si se cumple uno o más de los casos siguientes:

1. B agrega objetos de A
2. B contiene objetos de A
3. B registra instancias de objetos de A
4. B utiliza más estrechamente objetos de A.
5. B tiene datos de inicialización que se pasarán a un objeto de A cuando sea creado (por tanto, B es un Experto con respecto a la creación de A). B es un creador de los objetos A.

Un ejemplo de este patrón lo constituye la clase Configuración.

Patrón Bajo Acoplamiento: ¿Cómo soportar bajas dependencias, bajo impacto del cambio e incremento de la reutilización?

Debe haber pocas dependencias entre las clases. Si todas las clases dependen de todas ¿cuánto software podemos extraer de un modo independiente y reutilizarlo en otro proyecto? Para determinar el nivel de acoplamiento de clases, son muy buenos los diagramas de colaboración de UML. Uno de los principales síntomas de un mal diseño y alto acoplamiento es una herencia muy profunda. Siempre hay que considerar las ventajas de la delegación respecto de la herencia. Ejemplo de este patrón lo constituyen las relaciones entre las clases que integran el paquete Navegación

Patrón Alta Cohesión: ¿Cómo mantener la complejidad manejable?

Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable. Ejemplos de una baja cohesión son clases que hacen demasiadas cosas. En todas las metodologías se considera la refactorización. Uno de los elemento a refactorizar son las clases saturadas de métodos. Ejemplos de buen diseño se producen cuando se crean los denominados “paquetes de servicio” o clases agrupadas por funcionalidades que son fácilmente reutilizables (bien por uso directo o por herencia). Como ejemplo de este patrón se encuentran las clases del paquete Navegación, encargadas de esta funcionalidad en cualquiera de sus variantes.

Patrón Controlador: ¿Quién debería ser el responsable de gestionar un evento de entrada al sistema?

Asignar una responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase que representa una de las opciones siguientes:

1. Representa el sistema global, dispositivo o subsistema.
2. Representa un caso de uso en el que tiene lugar el evento del sistema a menudo denominado <nombre del caso de uso> Manejador, <nombre del caso de uso> coordinador, <nombre del caso de uso> Sesión.

- Utilice la misma clase controlador para todos los eventos del sistema en el mismo escenario de caso de uso.
- Informalmente, una sesión es una instancia de una conversación con un actor. Las sesiones pueden tener cualquier duración, pero se organizan a menudo en función de casos de uso.

Como ejemplo de este patrón se encuentra la clase MAPri.

3.2.4. Estándar de codificación

Los estándares de codificación, también llamados estilos de programación o convenciones de código, no son más que convenios para escribir código fuente en ciertos lenguajes de programación. Estos estándares elevan la mantenibilidad del código, sirven como punto de referencia para los programadores, mantienen un estilo de programación y ayudan a mejorar el proceso de codificación, haciéndolo, entre otras cosas, mucho más eficiente.

Para la implementación del framework de MAPri, se decidió utilizar el estándar Adobe Open Source Coding Conventions (Convenciones de Código Fuente Abierto de Adobe). Este estándar es totalmente consistente con el estándar ECMAScript.

Algunas de las pautas que establece el estándar escogido son:

Nomenclatura

Escoger una nomenclatura adecuada es crucial para crear código que sea fácil tanto de usar como de entender.

- **Abreviaturas:** Las abreviaturas deben evitarse como regla general. Por ejemplo `calcularValorOptimo()` es mejor nombre de método que `calcValOpt()`. Ser explícitos es más importante que disminuir la cantidad de caracteres en el código. A pesar de esto, existen algunas abreviaturas estandarizadas que sí pueden usarse:

acc para accesibilidad, como en `botonAcc`
auto para automático, como en `autoLlenado`
eval para evaluar, como en `evalNavegacion`
info para información, como en `botonInfo`
num para número, como en `numNodos`
min para mínimo, como en `minY`
max para máximo, como en `maxY`
nav para navegación, como en `barraNav`

➢ **Acrónimos:** Los acrónimos deben ponerse en mayúscula o en minúscula totalmente, pero nunca una combinación de ambos (ejemplo: `SWF`, `swf`, pero nunca `Swf`). Solamente pueden usarse en minúscula totalmente cuando ellos mismos sean identificadores o se encuentren al inicio de un identificador.

➢ **Formación de palabras:** Cuando un identificador contenga múltiples palabras, se usa una combinación de mayúsculas y minúsculas (ejemplo: `MovieClip`, `cargadorMovieClip`) o también se pueden separar las palabras por un guión bajo o underscore (ejemplo: `Carga_de_Texto`).

➢ **Nombre de las Clases:** Los nombres de las clases se deben comenzar con letra mayúscula, al igual que el resto de los sustantivos que compongan dicho nombre, en cualquier otro caso, la palabra se escribe con letra minúscula en su totalidad (ejemplo: `Carga_de_Imagen`). Todas las palabras se separan con underscore.

➢ **Nombre de las Interfaces:** Los nombres de las interfaces deben seguir la misma nomenclatura de las clases, colocándoles delante la letra mayúscula I (ejemplo: `INavegacion`).

➢ **Nombre de las Constantes:** Los nombres de las constantes deben escribirse con letra mayúscula en su totalidad y se deben separar las palabras con underscore.

```
public static const FOO_BAR:String = "fooBar";
```

El nombre de la constante debe coincidir con el de su valor en caso de que este sea una cadena de caracteres (`String`).

➢ **Nombre de las Propiedades:** Los nombres de las propiedades (atributos) de la clase, deben comenzar con underscore, y los métodos de obtención (`get`) y modificación (`set`) de sus valores deben llamarse igual que las propiedades correspondientes, sólo que sin el guión bajo.

```
public var _sonidoActual:Sound;
public function get sonidoActual():Sound
{
    //Implementación
}
public function set sonidoActual(value:Sound):void { //Implementación }
```

En el caso de los métodos de modificación, el parámetro que contiene el nuevo valor siempre debe llevar el nombre de `value`.

- **Nombre de las Funciones:** Los nombres de las funciones (métodos) deben comenzar con letra minúscula, colocando en mayúscula la inicial de cada palabra siguiente que lo componga.

```
public function reproducirSonido():void
```

La única excepción de esta regla son los constructores ya que deben llamarse exactamente igual que la clase.

Usos del Lenguaje:

En esta sección se tratará el uso de los elementos de ActionScript, ya que existen varias formas de expresar lo mismo.

- **Declaración de Tipos de Datos:** Se debe escribir un tipo de dato para cada constante, variable, parámetro y valor de retorno de las funciones. En caso de que se desee indicar que el elemento es sin tipo de datos, se debe expresar mediante un asterisco (ejemplo: `var valorVariable:*`).

- **Declaración de Arreglos y Objetos:** Para declarar arreglos y objetos, no debe utilizarse la palabra reservada new, sino que se deben emplear los literales `[]` y `{}`.

```
var arregloCanciones:Array = [];
var objetoCargaXML = {};
```

En el caso de los objetos, nunca deben llamarse objetos, ya que es algo sumamente genérico y dificultaría la capacidad de mantenimiento y reutilización a largo plazo del producto final.

- **Sentencias if:** Si existen varias sentencias condicionales que ejecuten una sola línea de código, estas líneas no deben incluirse entre llaves, pero si al menos una condicional ejecuta más de una línea de código, se deben encerrar entre llaves todas las líneas de código que ejecute cada función.

```
if (condicion1)
{
    //Implementación
}
else
{
    //Línea de código 1
    //Línea de código 2
}
```

Siempre se debe dejar un espacio entre la palabra reservada `if` y el paréntesis izquierdo, al igual que entre los operadores que se encuentran dentro de la condicional.

- **Sentencias for:** Las sentencias que se encuentren dentro de un ciclo, deben colocarse siempre dentro de llaves, incluso si es una sola línea de código. Si la variable límite del ciclo es una función que no necesita ser reevaluada en cada iteración, debe crearse una variable llamada n justo encima del ciclo e igualarse a esta función. Siempre se debe dejar un espacio entre la palabra reservada `for` y el paréntesis izquierdo, al igual que entre los elementos del ciclo y los operadores que se encuentran dentro del mismo.

```
var n:int = arregloCanciones.length;
for (var i:int = 0; i < n; i++)
{
    //Implementación
}
```

En el caso de los ciclos anidados la variable de control del ciclo exterior se debe llamar i y a la variable límite n, mientras que en el ciclo interior se deben nombrar j y m respectivamente.

- **Sentencias while:** Las líneas de código que se encuentran dentro de una sentencia de este tipo deben ser encerradas entre llaves.

```
while (i < n) { //Implementación }
```

Siempre se debe dejar un espacio entre la palabra reservada `while` y el paréntesis izquierdo, al igual que entre los operadores que se encuentran dentro de la sentencia.

- **Sentencias switch:** En las sentencias de este tipo, se debe encerrar cada cláusula de la `switch` entre llaves (bloque de código). Siempre se debe poner la palabra reservada `break` o `return` (en caso de devolver algún valor) dentro de ese bloque de código y nunca se deben poner estas dos palabras juntas en un mismo bloque.

```
switch (n)
{
    case 0:
    {
        //Implementación
        break;
    }
    case 1:
    {
        //Implementación
        return;
    }
    default:
    {
        //Implementación
        break;
    }
}
```

Se debe dejar siempre un espacio entre la palabra reservada `switch` y el paréntesis izquierdo, al igual que entre los operadores que se encuentran dentro de la sentencia.

3.3. Desarrollo de las iteraciones

Durante la fase de Planificación se detallaron las HU correspondientes a cada una de las iteraciones a desarrollar, teniendo en cuenta las prioridades y restricciones de tiempos previstas por el cliente. Para

darle cumplimiento a cada HU, primeramente se realiza una revisión del plan de iteraciones y de ser necesario se realizan modificaciones. Dentro del contenido de este plan se descomponen las HU en tareas de programación o ingeniería, asignándole de esta forma un equipo de desarrollo (o una persona) que será el responsable de su implementación. Las tareas no tienen que necesariamente ser entendidas por el cliente, pueden ser escritas en lenguajes técnicos, pues las mismas son usadas únicamente por los programadores.

3.3.1. Iteración 1

En esta iteración se le dio cumplimiento a la implementación a las HU que se consideraron de mayor importancia para el desarrollo del framework, con el fin de obtener una versión del producto con algunas de las funcionalidades críticas, que se verán reflejadas en los dos primeros modelos de desarrollo.

Tabla 3.3.1 HU abordadas en la primera iteración

Historias de Usuario	Tiempo de Implementación (semanas)	
	Estimación	Real
Actividad Interactiva	2	2
Carga de Configuración	1	0.5
Navegación	1	1
Fondo Musical	1	0.7
Carga de Medias	1	0.8

Tabla 3.3.2 1ª Tarea de la HU #-1

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 1
Nombre de la tarea: Creación de Actividad Interactiva Sopa de Palabras	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 7/01/2009	**Fecha fin:** 10/01/2009
Programador responsable: Yoennis Garrido Vargas	
Descripción: Implementación de las clases que componen la actividad interactiva Sopa de Palabras.	

Tabla 3.3.3 2ª Tarea de la HU #-1

Tarea de Ingeniería	
No. De la Tarea: 2	**No. De la HU:** 1
Nombre de la tarea: Creación de Actividad Interactiva Crucigrama	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 11/01/2009	**Fecha fin:** 13/01/2009
Programador responsable: Yoennis Garrido Vargas	
Descripción: Implementación de las clases que componen la actividad interactiva Crucigrama.	

Tabla 3.3.4 3ª Tarea de la HU #-1

Tarea de Ingeniería	
No. De la Tarea: 3	**No. De la HU:** 1
Nombre de la tarea: Creación de Actividad Interactiva Selección Simple	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 13/01/2009	**Fecha fin:** 14/01/2009
Programador responsable: Yoennis Garrido Vargas	
Descripción: Implementación de las clases que componen la actividad interactiva Selección Simple.	

Tabla 3.3.5 4ª Tarea de la HU #-1

Tarea de Ingeniería	
No. De la Tarea: 4	**No. De la HU:** 1
Nombre de la tarea: Creación de Actividad Interactiva Completamiento de Oraciones	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 14/01/2009	**Fecha fin:** 16/01/2009
Programador responsable: Yoennis Garrido Vargas	
Descripción: Implementación de las clases que componen la actividad interactiva Completamiento de Oraciones.	

Tabla 3.3.6 5ª Tarea de la HU #-1

Tarea de Ingeniería	
No. De la Tarea: 5	**No. De la HU:** 1
Nombre de la tarea: Creación de Actividad Interactiva Pareo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 16/01/2009	**Fecha fin:** 17/01/2009
Programador responsable: Yoennis Garrido Vargas	
Descripción: Implementación de las clases que componen la actividad interactiva Pareo.	

Tabla 3.3.7 1ª Tarea de la HU #-2

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 2
Nombre de la tarea: Implementación de la clase Configuración	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 05/01/2009	**Fecha fin:** 07/01/2009
Programador responsable: Fabian Finalé Franqui	
Descripción: Implementación de la clase configuración.	

Tabla 3.3.8 2ª Tarea de la HU #-2

Tarea de Ingeniería	
No. De la Tarea: 2	**No. De la HU:** 2
Nombre de la tarea: Implementación de la clase Ruta	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 08/01/2009	**Fecha fin:** 10/01/2009
Programador responsable: Fabian Finalé Franqui	
Descripción: Implementación de la clase Ruta.	

Tabla 3.3.9 1ª Tarea de la HU #-3

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 3
Nombre de la tarea: Creación del paquete Navegación	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 12/01/2009	**Fecha fin:** 19/01/2009
Programador responsable: Fabian Finalé Franqui	
Descripción: Implementación de las clases que componen al paquete Navegación.	

Tabla 3.3.10 1ª Tarea de la HU #-4

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 4
Nombre de la tarea: Creación de la carga del Fondo Musical	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 21/01/2009	**Fecha fin:** 26/01/2009
Programador responsable: Fabian Finalé Franqui	
Descripción: Implementación de las clases que componen el fondo musical.	

Tabla 3.3.11 1ª Tarea de la HU #-5

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 5
Nombre de la tarea: Creación de las clases que cargan medias.	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 28/01/2009	**Fecha fin:** 3/02/2009
Programador responsable: Fabian Finalé Franqui	
Descripción: Implementación de las clases que cargan las medias.	

3.3.2. Iteración 2

En esta iteración se implementaron las HU que responden a los números 6 y 7. Dichas HU son las que brindan las funcionalidades de reproducción de contenidos y visualización de imágenes a través de una galería de imágenes.

Tabla 3.3.12 HU abordadas en la segunda iteración

Historias de Usuario	Tiempo de Implementación (semanas)	
	Estimación	Real
Reproductor	2	1.5
Galería de Imágenes	1	1

Tabla 3.3.13 1ª Tarea de la HU #-6

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 6
Nombre de la tarea: Creación del Reproductor	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 08/02/2009	**Fecha fin:** 18/02/2009
Programador responsable: Fabian Finalé Franqui	
Descripción: Implementación de las clases que complementan el reproductor.	

Tabla 3.3.14 1ª Tarea de la HU #-7

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 7
Nombre de la tarea: Creación de la Galería	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 20/02/2009	**Fecha fin:** 27/02/2009
Programador responsable: Fabian Finalé Franqui	
Descripción: Implementación de las clases que componen la galería.	

3.3.3. Iteración 3

En esta última iteración del framework, se les dio cumplimiento a las HU 8 y 9. Dichas HU cumplen con las funcionalidades de impresión de documentos y realización de búsquedas de información referente a algún tema de interés para los usuarios.

Tabla 3.3.15 HU abordadas en la tercera iteración

Historias de Usuario	Tiempo de Implementación (semanas)	
	Estimación	Real
Impresión de Archivos	2	1.8
Buscador	2	2

Tabla 3.3.16 1ª Tarea de la HU #-8

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 8
Nombre de la tarea: Creación de la clase Impresión	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 1/04/2009	**Fecha fin:** 13/04/2009
Programador responsable: Fabian Finalé Franqui	
Descripción: Implementación de la clase que permite imprimir un documento.	

Tabla 3.3.17 1ª Tarea de la HU #-9

Tarea de Ingeniería	
No. De la Tarea: 1	**No. De la HU:** 9
Nombre de la tarea: Implementación del Buscador de Contenidos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 2
Fecha inicio: 30/04/2009	**Fecha fin:** 10/05/2009
Programador responsable: Yoennis Garrido Vargas	
Descripción: Implementación de las clases que componen el Buscador de Contenidos.	

3.4. Pruebas

En la metodología XP es esencial el desarrollo de las pruebas, permitiendo probar constantemente el código. Cada vez que se quiere implementar las funcionalidades que tendrá el software, XP propone una redacción sencilla de prueba, para ser pasada por el código posteriormente. El desarrollo constante de las pruebas permite que se desarrolle un producto con mayor calidad dando a los programadores una mayor seguridad en el trabajo que desarrollan. En XP hay dos tipos de pruebas; las unitarias o TDD (desarrollo dirigido por pruebas, del inglés *test driven development*), desarrolladas por los programadores verificando su código de forma automática y las pruebas de aceptación, las cuales son evaluadas luego de culminar una iteración verificando así si se cumplió la funcionalidad requerida por el cliente. Con estas normas se obtiene un código simple y funcional de manera bastante rápida. Por esto es importante pasar las pruebas al 100%.

3.4.1. Desarrollo Dirigido por Pruebas

El desarrollo dirigido por pruebas (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, pero sí funcional. Algunas personas confunden este término con las nombradas “pruebas de caja blanca”, las cuales son pruebas que se realizan a los métodos u operaciones para medir la funcionalidad del mismo desde la perspectiva de la validez para el cliente. Sin embargo el TDD 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.

Debido a que ActionScript no cuenta con un framework que permita realizar las pruebas unitarias o el TDD, se hace necesario utilizar el depurador de Flex Builder o la consola de salida en última instancia.

3.4.2. Pruebas de Aceptación

Las pruebas de aceptación en la metodología XP, se pueden asociar con las pruebas de caja negra que se aplican en otras metodologías de desarrollo, sólo que en XP se crean a partir de las historias de

usuario y no por un listado de requerimientos. Durante las iteraciones las historias de usuarios seleccionadas serán traducidas a pruebas de aceptación. En ellas se especifican, desde la perspectiva del cliente, los escenarios para probar que una HU ha sido implementada correctamente. Una HU puede tener todas las pruebas de aceptación que necesite para asegurar su correcto funcionamiento. El objetivo final de éstas es garantizar que las funcionalidades requeridas por el cliente han sido cumplidas. Una HU no se considera completa hasta que no ha pasado por sus pruebas de aceptación.

Tabla 3.4.1 Prueba de aceptación para la HU "Actividad Interactiva"

Caso de Prueba de Aceptación	
Código: HU1_P1	**Historia de Usuario:** 1
Nombre: Actividades Interactivas	
Descripción: Prueba para la funcionalidad de las Actividades Interactivas.	
Condiciones de Ejecución: El cliente debe probar que todas las Actividades Interactivas requeridas, cumplan con la expectativa esperada.	
Entrada/ Pasos de ejecución: Se solicita la Actividad Interactiva deseada y posteriormente se procede a probar que dicha actividad funcione correctamente.	
Resultado Esperado: Las Actividades Interactivas no presentan errores.	
Evaluación de la Prueba: -	

Tabla 3.4.2 Prueba de aceptación para la HU "Carga de Configuración"

Caso de Prueba de Aceptación	
Código: HU2_P1	**Historia de Usuario:** 2
Nombre: Carga de Configuración	
Descripción: Prueba para la funcionalidad de la Carga de Configuración.	
Condiciones de Ejecución: El cliente debe modificar el archivo de configuración.	
Entrada/ Pasos de ejecución: Una vez modificado el archivo, se procede a revisar en la nueva aplicación multimedia todos los elementos establecidos por el cliente.	
Resultado Esperado: Se genera una aplicación multimedia acorde a la configuración del archivo modificado.	
Evaluación de la Prueba: -	

Tabla 3.4.3 Prueba de aceptación para la HU "Navegación"

Caso de Prueba de Aceptación	
Código: HU3_P1	**Historia de Usuario:** 3
Nombre: Navegación	
Descripción: Prueba para comprobar la navegación.	
Condiciones de Ejecución: El cliente debe comprobar que la aplicación multimedia navegue acorde a lo establecido en el archivo de configuración.	
Entrada/ Pasos de ejecución: Se navega por la aplicación multimedia, probando todos los botones de navegación.	
Resultado Esperado: Cada botón realiza su función correspondiente.	
Evaluación de la Prueba: -	

Tabla 3.4.4 Prueba de aceptación para la HU "Fondo Musical"

Caso de Prueba de Aceptación	
Código: HU4_P1	**Historia de Usuario:** 4
Nombre: Fondo Musical	
Descripción: Prueba para revisar la funcionalidad del fondo musical de la aplicación multimedia.	
Condiciones de Ejecución: El cliente debe comprobar que el fondo musical opere de acuerdo a lo establecido en el archivo de configuración.	
Entrada/ Pasos de ejecución: Se pausa y se reproduce el fondo musical. Se analiza si los sonidos de fondo se reproducen correctamente.	
Resultado Esperado: Cada botón realiza su función correspondiente y la reproducción del fondo musical se realiza de acuerdo a lo establecido en el archivo de configuración.	
Evaluación de la Prueba: -	

Tabla 3.4.5 Prueba de aceptación para la HU "Carga de Medias"

Caso de Prueba de Aceptación	
Código: HU5_P1	**Historia de Usuario:** 5
Nombre: Carga de Medias	
Descripción: Prueba que determina el funcionamiento de la carga de medias.	
Condiciones de Ejecución: El cliente debe cargar los distintos tipos de media.	
Entrada/ Pasos de ejecución: Se modifica el archivo *.xml de una de las páginas, estableciendo la carga de los distintos tipos de medias.	
Resultado Esperado: Todas los tipos de medias son cargadas correctamente.	
Evaluación de la Prueba: -	

Tabla 3.4.6 Prueba de aceptación para la HU "Reproductor"

Caso de Prueba de Aceptación	
Código: HU6_P1	**Historia de Usuario:** 6
Nombre: Reproductor	
Descripción: Evalúa la funcionalidad del reproductor.	
Condiciones de Ejecución: El cliente debe probar todas las funcionalidades del reproductor.	
Entrada/ Pasos de ejecución: Todas las funcionalidades del reproductor son probadas.	
Resultado Esperado: Se reproducen correctamente los videos, archivos *.swf y archivos *.mp3.	
Evaluación de la Prueba: -	

Tabla 3.4.7 Prueba de aceptación para la HU "Galería de Imágenes"

Caso de Prueba de Aceptación	
Código: HU7_P1	**Historia de Usuario:** 7
Nombre: Galería de Imágenes	
Descripción: Prueba la funcionalidad de la galería de imágenes.	
Condiciones de Ejecución: El cliente debe probar y visualizar todas las imágenes de la galería.	
Entrada/ Pasos de ejecución: Se navega a través de la galería de imágenes y se visualizan todas las imágenes presionando sus vistas en miniatura.	
Resultado Esperado: La navegación a través de la galería se realiza correctamente y al presionar una vista en miniatura la imagen se visualiza.	
Evaluación de la Prueba: -	

Tabla 3.4.8 Prueba de aceptación para la HU "Impresión de Archivos"

Caso de Prueba de Aceptación	
Código: HU8_P1	**Historia de Usuario:** 8
Nombre: Impresión de archivos	
Descripción: Prueba la funcionalidad de la impresión de archivos.	
Condiciones de Ejecución: El cliente debe imprimir un archivo determinado.	
Entrada/ Pasos de ejecución: Se selecciona un elemento y se presiona el botón de impresión.	
Resultado Esperado: Se imprime el elemento.	
Evaluación de la Prueba: -	

Tabla 3.4.9 Prueba de aceptación para la HU "Buscador"

Caso de Prueba de Aceptación	
Código: HU9_P1	**Historia de Usuario:** 9
Nombre: Buscador de Contenidos	
Descripción: Prueba para la funcionalidad del buscador de contenidos. Probar que busque correctamente la palabra o texto que se desee buscar.	
Condiciones de Ejecución: El criterio de búsqueda debe ser ingresado correctamente con la palabra o texto que se desee buscar.	
Entrada/ Pasos de ejecución: Se introduce la palabra o texto que se desee buscar.	
Resultado Esperado: En caso de que la palabra o texto que se desee buscar se encuentre en los contenidos o al menos en alguno de ellos, se le debe mostrar el listado de las referencias donde se encuentra dicha palabra o texto que se desee buscar. En caso de que no se encuentre lo que se desea buscar, se les debe informar a los usuarios que no se ha encontrado la palabra o texto que se desea buscar.	
Evaluación de la Prueba: -	

3.5. Conclusiones

En este capítulo se construyó el diagrama de clases del framework, logrando una visión detallada de sus atributos y las relaciones entre ellas. Se elaboraron las tarjetas CRC y se definió el estándar de codificación a utilizar. Se desarrollaron las tareas correspondientes para dar solución a las historias de usuario y se definió el modelo arquitectónico a seguir, cumpliendo así el objetivo de este trabajo: desarrollar una arquitectura para la creación de aplicaciones multimedia.

CONCLUSIONES

MAPri brinda una solución factible a la problemática inicial y su desarrollo significará una mejora considerable en el desarrollo de productos con tecnología multimedia en la Universidad. Al finalizar el desarrollo de la arquitectura, se dio cumplimiento satisfactoriamente a los objetivos establecidos inicialmente:

- Se logró integrar la arquitectura a los estándares de creación de aplicaciones multimedia.
- Se implementó el framework de la arquitectura estableciendo los paquetes necesarios y trazando las pautas para el futuro desarrollo de MAPri.
- Se implementaron los modelos de desarrollo con todas las funcionalidades esenciales de las aplicaciones multimedia.
- Se elaboró la documentación para las guías de desarrollo de la arquitectura, estableciendo así las instrucciones para el uso de MAPri en la creación de nuevos productos.
- Se confeccionaron los ejemplos en formato *.swf para mostrar las posibilidades del trabajo con MAPri.

Se puede concluir que el desarrollo de la Micro Arquitectura Primavera presenta una solución al desarrollo multimedia en la Universidad de las Ciencias Informáticas, constituyendo un nuevo enfoque para la producción de aplicaciones de este tipo. Elaborar aplicaciones multimedia en cuestión de días a partir de la entrega total de la información del cliente, ya no es un sueño.

RECOMENDACIONES

Tomando como punto de partida los resultados obtenidos con la realización de este trabajo de diploma, se hacen las siguientes recomendaciones:

- Aplicar la Micro-Arquitectura Primavera en el desarrollo de los productos con tecnología multimedia en la Universidad de las Ciencias Informáticas.
- Continuar el desarrollo del framework de la arquitectura para implementar nuevas funcionalidades e incorporarlas a los Modelos de Desarrollo.
- Migrar el framework a ActionScript 2.0, HAXE, JavaScript, etc.; para posibilitar el uso de MAPri en otros proyectos desarrollados en la Universidad de las Ciencias Informáticas que utilicen esos lenguajes de programación.

REFERENCIAS BIBLIOGRÁFICAS

Alejos, A.A. 2007. Conocimientos - La divisa del nuevo milenio - ¿Qué es multimedia? [En línea] 2007. http://www.conocimientosweb.net/portal/article36.html.

Álvarez, Solís y Díaz, Figueroa. 2007. Metodologías Tradicionales vs. Metodologías Ágiles. [En línea] 2007. http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles.1515.

Balkan, A. 2004. Ariaware RIA Platform. [En línea] 2004. http://www.ariaware.com/products/arp/docs/arp.htm.

Clinger, A. 2007. CASA Lib: An Open Source Code Library for ActionScript 2.0 & 3.0. [En línea] 2007. http://casalib.org/.

Colectivo de Autores del INSTED. 2006. Toolbook. [En línea] 2006. http://www.insted.rimed.cu/documentos/Cap2.pdf.

FlashDevelop Development Team. 2009. Features - FlashDevelop. [En línea] 2009. http://www.flashdevelop.org/wikidocs/index.php?title=Features.

Grossman, G. y Huang, E. 2006. ActionScript 3.0 overview. [En línea] 2006. http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html.

Jorquera, J. 2008. Introducción a los Métodos Ágiles. [En línea] 2008. http://www.inf.udec.cl/~eossesa/DSIS/docs/Agiles%20Koke.ppt..

Letelier, P. 2006. Proceso Unificado de Desarrollo (RUP). [En línea] 2006. https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducci%C3%B3n%20a%20RUP.doc.

Letelier, P. 2006. CyTA - Crystal. [En línea] 2006. http://www.cyta.com.ar/ta0502/v5n2a1.htm#3.

Letelier, P. y Penadés, M.d.C. 2006. eXtreme Programming (XP). [En línea] 2006. http://www.willydev.net/descargas/masyxp.pdf.

Mony, T. 2008. Crystal Methodologies. [En línea] 2008. http://crystalmethodologies.blogspot.com/.

Moreno Martínez, G. 2000. Rational Rose. [En línea] 2000. http://www.monografias.com/trabajos5/insof/insof.shtml.

Penadés, M.d.C. 2006. CyTA - SCRUM. [En línea] 2006. http://www.cyta.com.ar/ta0502/v5n2a1.htm.

Sacks, S. 2006. Gaia Framework for Adobe Flash. [En línea] 2006. http://www.gaiaflashframework.com/.

Stoica, L. 2007. Introducing VEGAS, an Actionscript 3 / 2 and SSAS framework - Global event-listeners and addEventListener for ALL events (Comtaste Consulting | Enterprise RIA consulting and development). [En línea] 2007. http://blog.comtaste.com/2007/09/introducing_vegas_an_actionscr.html.

The Cairngorm Team. 2008. Cairngorm - Cairngorm - Confluence. [En línea] 2008. http://opensource.adobe.com/wiki/display/cairngorm/Cairngorm.

Tynjala, J. 2008. SimpleAS3: An ActionScript Framework for Designers, Animators, and Part-time Coders - Josh Talks Flash. [En línea] 2008. http://joshblog.net/2008/08/13/simpleas3-an-actionscript-framework-for-designers-animators-and-part-time-coders/.

Universidad Politécnica de Valencia. 2004. CASE. [En línea] 2004. http://www.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf.

Visual Paradigm International. 2007. Visual Paradigm. [En línea] 2007. http://www.visual-paradigm.com.

Xavier, M. 2007. Dev by MX» AS3 Framework - lowRa. [En línea] 2007. http://dev.webbymx.net/2007/08/20/as3-framework-lowra/.

Zhang, M. 2008. InfoQ: ActionScript Framework JumpShip 3.1 Released with Extensive Ruby on Rails Support. [En línea] 2008. http://www.infoq.com/news/2008/06/jumpship-31-released.

BIBLIOGRAFÍA

Alejos, A.A. 2007. Conocimientos - La divisa del nuevo milenio - ¿Que es multimedia?

Álvarez, Solís y Díaz, Figueroa. 2007. Metodologías Tradicionales vs. Metodologías Ágiles.

Balkan, A. 2004. Ariaware RIA Platform. http://www.ariaware.com/products/arp/docs/arp.htm.

Clinger, A. 2007. CASA Lib: An Open Source Code Library for ActionScript 2.0 & 3.0. http://casalib.org/.

Colectivo de Autores del INSTED. 2006. Toolbook.

FlashDevelop Development Team. 2009. Features - FlashDevelop. http://www.flashdevelop.org/wikidocs/index.php?title=Features.

Grossman, G. y Huang, E. 2006. ActionScript 3.0 overview. http://www.adobe.com/devnet/actionscript/articles/actionscript3_overview.html.

Jorquera, J. 2008. Introducción a los Métodos Ágiles.

Letelier, P. 2006. Proceso Unificado de Desarrollo (RUP).

Letelier, P. 2006. CyTA - Crystal. http://www.cyta.com.ar/ta0502/v5n2a1.htm#3.

Letelier, P. y Penadés, M.d.C. 2006. eXtreme Programming (XP). http://www.willydev.net/descargas/masyxp.pdf.

Mony, T. 2008. Crystal Methodologies. http://crystalmethodologies.blogspot.com/.

Moreno Martínez, G. 2000. Rational Rose. http://www.monografias.com/trabajos5/insof/insof.shtml.

Penadés, M.d.C. 2006. CyTA - SCRUM. http://www.cyta.com.ar/ta0502/v5n2a1.htm.

Sacks, S. 2006. Gaia Framework for Adobe Flash. http://www.gaiaflashframework.com/.

Stoica, L. 2007. Introducing VEGAS, an Actionscript 3 / 2 and SSAS framework - Global event-listeners and addEventListener for ALL events (Comtaste Consulting | Enterprise RIA consulting and development). http://blog.comtaste.com/2007/09/introducing_vegas_an_actionscr.html.

The Cairngorm Team. 2008. Cairngorm - Cairngorm - Confluence. http://opensource.adobe.com/wiki/display/cairngorm/Cairngorm.

Tynjala, J. 2008. SimpleAS3: An ActionScript Framework for Designers, Animators, and Part-time Coders - Josh Talks Flash. http://joshblog.net/2008/08/13/simpleas3-an-actionscript-framework-for-designers-animators-and-part-time-coders/.

Universidad Politécnica de Valencia. 2004. CASE. http://www.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf.

Visual Paradigm International. 2007. Visual Paradigm. http://www.visual-paradigm.com.

Xavier, M. 2007. Dev by MX» AS3 Framework - lowRa. http://dev.webbymx.net/2007/08/20/as3-framework-lowra/.

Z**amora Sánchez, G. & Rodríguez Tamayo, Y**. Análisis de un IDE para múltiples plataformas con tecnologías y herramientas libres para desarrollar software educativo en formato multimedia. Análisis, diseño e implementación de componentes, ejercicios y clases reutilizables. Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba, 2008.

Zhang, M. 2008. InfoQ: ActionScript Framework JumpShip 3.1 Released with Extensive Ruby on Rails Support. http://www.infoq.com/news/2008/06/jumpship-31-released.

GLOSARIO DE TÉRMINOS

ActionScript: Lenguaje de programación orientado a objetos utilizado en especial en aplicaciones web animadas realizadas en el entorno Adobe Flash, la tecnología de Adobe para añadir dinamismo al panorama web. Fue lanzado con la versión 4 de Flash, y desde entonces hasta ahora, ha ido ampliándose poco a poco, hasta llegar a niveles de dinamismo y versatilidad muy altos en la versión 10 (Adobe Flash CS4) de Flash.

Adobe Flash Player: Aplicación en forma de reproductor multimedia creado inicialmente por Macromedia y actualmente distribuido por Adobe Systems. Permite reproducir archivos SWF que pueden ser creados con la herramienta de autoría Adobe Flash, con Adobe Flex o con otras herramientas de Adobe y de terceros.

Adobe Flash Professional: Aplicación en forma de estudio de animación que trabaja sobre fotogramas destinado a la producción y entrega de contenido interactivo para diferentes audiencias alrededor del mundo sin importar la plataforma. Es actualmente escrito y distribuido por Adobe Systems, y utiliza gráficos vectoriales e imágenes ráster (mapas de bits), sonido, código de programa, flujo de vídeo y audio bidireccional. En sentido estricto, Flash es el entorno y Flash Player es el programa de máquina virtual utilizado para ejecutar los archivos generados con Flash.

Aplicación Multimedia: Solución informática que usa simultáneamente diferentes formas de contenido informativo como textos, sonidos, imágenes y animaciones.

Aplicación: Tipo de programa informático diseñado como herramienta para permitir a un usuario realizar diversos tipos de trabajo. Suele resultar una solución informática para la automatización de ciertas tareas complicadas como pueden ser la contabilidad, la redacción de documentos, o la gestión de un almacén.

Arquitectura: Estructura de sistemas o sistema de estructuras que consisten en elementos, sus propiedades externamente visibles y la relación entre ellos. Representa las decisiones de diseño significativas que le dan forma a un sistema, donde lo significativo puede ser medido por el costo del cambio**.**

Automatización: uso de sistemas o elementos computarizados para controlar maquinarias y/o procesos industriales substituyendo a operadores humanos.

Clase: Contenedor de uno o más datos (variables o propiedades miembro) junto a las operaciones de manipulación de dichos datos (funciones/métodos).

Cliente de Subversion: Interfaz para el software de sistema de control de versiones Subversion, diseñado específicamente para reemplazar al popular CVS.

Código Abierto: Término con el que se conoce al software distribuido y desarrollado libremente. Fue utilizado por primera vez en 1998 por algunos usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original en inglés del software libre (*free software*).

Código Fuente: Conjunto de líneas de texto indicando las instrucciones que debe seguir un ordenador para ejecutar un programa, por lo que en el código fuente de un programa está descrito por completo su funcionamiento. El código fuente está escrito por un programador en algún lenguaje de programación, pero en este primer estado no es directamente ejecutable por el ordenador, sino que debe ser traducido a otro lenguaje (el lenguaje máquina o código objeto) que sí pueda ser ejecutado por el hardware de la computadora. Para esta traducción se usan los llamados compiladores, ensambladores, intérpretes y otros sistemas de traducción.

Desarrollo Multimedia Web: Actividades y estudios que permitan e incentiven el desarrollo tanto de la tecnología multimedia como de la web.

ECMA International: Organización internacional basada en membrecías de estándares para la comunicación y la información. Adquirió el nombre *ECMA International* en 1994, cuando la *European Computer Manufacturers Association* (ECMA), cambió su nombre para expresar su alcance internacional. Fue fundada en 1961 para estandarizar los sistemas computarizados en Europa.

ECMAScript: Especificación de lenguaje de programación publicada por ECMA International. Su desarrollo empezó en 1996 y estuvo basado en el popular lenguaje JavaScript propuesto como estándar por Netscape Communications Corporation. Actualmente está aceptado como el estándar ISO 16262.

Editor de Código ActionScript: Editor de código diseñado específicamente para trabajar con ActionScript como lenguaje de programación.

Editor de Código: Editor de texto diseñado específicamente para escribir por los programadores el código fuente de alguna aplicación. Puede ser una aplicación independiente o estar incorporada a un entorno de desarrollo integrado (IDE por sus siglas en inglés). Los editores de código varían acorde al lenguaje de programación que permitan utilizar.

Encapsulación: Ocultamiento del estado o de los datos de un objeto de manera que sólo se puedan cambiar mediante las operaciones definidas para ese objeto. Protege a los datos asociados a un objeto contra su modificación por quien no tenga derecho a acceder a ellos, eliminando efectos secundarios e interacciones.

Estándar: acuerdo documentado que contiene especificaciones técnicas u otros criterios específicos para ser usado como referencia, guía o definición de características, para asegurar qué materiales, productos, procesos y servicios son obtenidos o han sido realizados de acuerdo a sus propósitos.

Factoría de Software: estructura organizacional que se especializa en la producción de software o componentes de software acorde a requerimientos del usuario final externamente definidos a través del proceso de ensamblaje.

Fichero: Conjunto de información que se almacena en algún medio de escritura que permita ser leído o accedido por una computadora. Es identificado por un nombre y la descripción de la carpeta o directorio que lo contiene y facilita una manera de organizar los recursos usados para almacenar permanentemente información dentro de un computador.

Formato *.swf: Formato de archivo de gráficos vectoriales creado por la empresa Macromedia (actualmente Adobe Systems). Los archivos con este formato pueden ser creados con Adobe Flash, aunque hay otras aplicaciones que también lo permiten. Básicamente es un formato vectorial, aunque también admite mapas de bits. Necesita para ser ejecutado el reproductor Adobe Flash Player, el cual permite mostrar las animaciones vectoriales que contienen los ficheros. Los archivos SWF suelen ser suficientemente pequeños como para ser publicados en la web en forma de animaciones o componentes (applets) con diversas funciones y grados de interactividad. También son usados frecuentemente para

crear animaciones y gráficos en otros medios, como menús para películas en DVD y anuncios de televisión.

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

Gestor de Contenido: Aplicación utilizada para crear, editar, administrar, buscar y publicar varios los distintos tipos de media existentes.

GNU/Linux: Término empleado para referirse al sistema operativo similar a Unix que utiliza como base las herramientas de sistema de GNU y el núcleo Linux. Su desarrollo es uno de los ejemplos más prominentes de software libre; todo el código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL de GNU y otras licencias libres.

Google Code: Sitio web de Google para desarrolladores que contiene códigos de fuente abiertos y una lista de los servicios soportados por las APIs públicas.

HaXe: Lenguaje de programación diseñado para crear aplicaciones web interactivas. Está orientado a cuatro máquinas virtuales o compiladores: Adobe Flash, JavaScript, PHP y el Neko VM.

Herramienta de Ingeniería de Software Asistida por Ordenador (CASE): Aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero.

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

Instalador: Herramienta que permite copiar y configurar automáticamente una aplicación en un ordenador para que pueda ser ejecutada.

Interfaz de Programación de Aplicaciones (API): Conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Interfaz: Parte de un programa que permite el flujo de información entre un usuario y la aplicación, o entre la aplicación y otros programas o periféricos. Está constituida por un conjunto de comandos y métodos que permiten estas intercomunicaciones. Puede ser de tipo gráfico (GUI por sus siglas en inglés) o de modo consola.

Internet: Conjunto descentralizado de redes de comunicación interconectadas, que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial.

Interoperabilidad: Propiedad mediante la cual los sistemas pueden intercambiar procesos o datos.

ISO Open Document: Formato de fichero estándar para el almacenamiento de documentos ofimáticos tales como hojas de cálculo, memorandos, gráficas y presentaciones.

Iteración: Técnica de desarrollar y entregar componentes incrementales de funcionalidades de un negocio.

Lenguaje de marcado extensible (XML): Metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos de la misma manera que HTML. Por lo tanto XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

Lenguaje de marcado extensible multimedia (MXML): Lenguaje descriptivo desarrollado inicialmente por Macromedia hasta el 2005 para la plataforma Flex de Adobe. Describe interfaces de usuario, crea modelos de datos y tiene acceso a los recursos del servidor. Tiene una mayor estructura en base a etiquetas, similar a HTML, pero con una sintaxis menos ambigua, proporciona una gran variedad e inclusive permite extender etiquetas y crear sus propios componentes.

Lenguaje de Marcas de Hipertexto (HTML): Lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto y complementar ese texto con imágenes y objetos de diversos tipos. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un

documento y puede afectar el comportamiento de navegadores web y otros procesadores de HTML mediante lenguajes de programación.

Lenguaje de Programación: Conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina.

Lenguaje Unificado de Modelado (UML): 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.

Licencia BSD: Licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Pertenece al grupo de licencias de software Libre. Esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público. La licencia BSD al contrario que la GPL permite el uso del código fuente en software no libre. Es muy similar en efectos a la licencia MIT.

Licencia GNU/GPL: Licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Licencia MIT: Una de tantas licencias de software que ha empleado el Instituto de Tecnología de Massachusetts (MIT por sus siglas en inglés) a lo largo de su historia, y debería llamarse licencia X11, ya que fue empleada por primera vez en X Window System en los años 80.

Mac OS-X: Línea de sistemas operativos computacionales desarrollada, comercializada y vendida por Apple Incorporated. Se basa en Unix y usa una interfaz gráfica desarrollada por Apple llamada Aqua, que se inspira libremente en la interfaz de Mac OS Classic.

Metodología de Desarrollo: Marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Metodología: Métodos de investigación que se siguen para alcanzar una gama de objetivos en una ciencia.

Métodos: Porciones de código asociadas exclusivamente a una clase o a un objeto. Análogamente a los procedimientos en los lenguajes imperativos, un método consiste generalmente de una serie de sentencias para llevar a cabo una acción, un juego de parámetros de entrada que regularán dicha acción y, posiblemente, un valor de salida (o valor de retorno) de algún tipo. El propósito de los métodos es el de proveer un mecanismo para acceder (leer o modificar) los datos privados que se encuentran almacenados en un objeto o clase.

Microsoft Windows: Familia de sistemas operativos desarrollados y comercializados por Microsoft. Existen versiones para hogares, empresas, servidores y dispositivos móviles, como computadores de bolsillo y teléfonos inteligentes. Hay variantes para procesadores de 16, 32 y 64 bits.

Motor de Renderizado Gecko: Motor de renderizado libre escrito en C++ que toma el contenido (HTML, XML, imágenes, etc.) y la información de formato (CSS, etc.) y crea una representación visual de una página web. Fue originalmente desarrollado por Netscape y actualmente su desarrollo es gestionado por la Fundación Mozilla y la Corporación Mozilla.

Navegación por Pestañas: Forma de navegación donde varios paneles con información están contenidos dentro de una sola ventana principal, usando pestañas para alternar entre ellos.

Página Web: documento HTML/XHTML accesible generalmente mediante el protocolo HTTP de Internet.

Paradigma de Programación: Enfoque particular o filosofía para la construcción del software. No es mejor uno que otro sino que cada uno tiene ventajas y desventajas. También hay situaciones donde un paradigma resulta más apropiado que otro.

Patrón Inversión de Control: Patrón de diseño donde el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos o funciones. Tradicionalmente el programador especifica la secuencia de decisiones y procedimientos que pueden darse durante el ciclo de vida de un programa mediante llamadas a funciones. En su lugar, en la inversión de control se especifican respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que algún tipo de entidad o arquitectura

externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.

Patrón Modelo Vista Controlador: Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Patrón Singleton: Patrón diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.

Plataforma Flash: aplicación en forma de estudio de animación que trabaja sobre *"Fotogramas"* destinado a la producción y entrega de contenido interactivo para diferentes audiencias alrededor del mundo.

Plug-in: Aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.

Programación Orientada a Objetos: Paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento.

Restricciones Legales: Todas aquellas leyes que prohíben algún aspecto de determinado software, ya sea la reproducción, modificación o distribución.

Sitio Web: Conjunto de páginas web, típicamente comunes a un dominio o subdominio de Internet.

Software Educativo: Software destinando a la enseñanza y el auto aprendizaje y que además permite el desarrollo de ciertas habilidades cognitivas.

Software Privativo: Software en el que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo (con o sin modificaciones), o cuyo código fuente no está disponible o el acceso a éste se encuentra restringido.

Software: Equipamiento lógico o soporte lógico de un ordenador. Comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware). Suele ser utilizado para referirse a los programas y aplicaciones informáticas.

Solaris: Sistema operativo de tipo Unix desarrollado por Sun Microsystems desde 1992 como sucesor de SunOS. Es un sistema certificado oficialmente como versión de Unix. Funciona en arquitecturas SPARC y x86 para servidores y estaciones de trabajo. Aunque Solaris fue desarrollado como software privativo, la mayor parte de su código se ha liberado como proyecto de software libre denominado *OpenSolaris*.

Sprint: Período de tiempo en el desarrollo de software enfocado en una lista de metas. Es la unidad básica de desarrollo de las metodologías ágiles y su duración tiende a oscilar entre una semana y un mes.

Streaming: Término que se refiere a ver u oír un archivo directamente en una página web sin necesidad de descargarlo antes al ordenador. En términos más complejos podría decirse que describe una estrategia sobre demanda para la distribución de contenido multimedia a través del internet.

Subversion: Software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como svn por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Suite Ofimática: Conjunto de aplicaciones para el uso en oficinas y entornos profesionales. Aunque no hay un estándar sobre los programas a incluir, la mayoría incluye al menos un procesador de texto y una hoja de cálculo. De forma añadida, la suite puede contener un programa de presentación, un sistema de gestión de base de datos, herramientas menores de gráficos y comunicaciones, un gestor de información personal (agenda y cliente de correo electrónico) y un navegador web.

Tecnologías de la Información y las Comunicaciones: conjunto de servicios, redes, software y dispositivos que tienen como fin la mejora de la calidad de vida de las personas dentro de un entorno, y que se integran a un sistema de información interconectado y complementario.

Unix: Sistema operativo portable, multitarea y multiusuario desarrollado en 1969 por un grupo de empleados de los laboratorios Bell de AT&T.

Anexo 1. Diagrama de Clases del framework.