Universidad de las Ciencias Informáticas
Facultad 4

Título: Metodologías para el desarrollo de software multimedia: Análisis comparativo y propuesta

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Kenia Labori de la Rosa

Izary Rondón Robaúl

Tutor: Ing. Rolando Quintana Aput

Co-tutor: Ing. Abel Meneses Abad

Consultante: MSc. Rafael Barrera Yanes

Junio, 2007

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Izary Rondón Robaúl

_____

Autora

Kenia Labori de la Rosa

_____

Autora

Rolando Quintana Aput

_____

Tutor

AGRADECIMIENTOS

Son muchas las personas a las que tenemos que agradecer la realidad de la tesis.

De Izary:

A mis padres, Herminia y Genaro, por su inmenso apoyo al cumplimiento de mi sueño, a mis hermanos Victoria, Olga, Odalys y Carlos, por sus siempre acertados consejos y su paciencia. A mi familia en general, por siempre estar ahí, en lugar y en el momento correcto.

A Abel, por su orientación en todo el trabajo.

A todos los que han tenido que ver con mi vida de estudiante en la universidad, a esos que tanto influyeron en mi formación, tanto lo personal como en lo profesional, comenzando por mis compañeras y amigas Kenia y Yani que tan amena me hicieron la estancia allí y que siempre estuvieron conmigo, a los que se han ido por una u otra razón, Frances, Raidis, Saily, incluso a los que han venido llegando, Lianet.

De Kenia:

A mi mamá Isabel y a mi hermano Sandro, por su amor y su apoyo constante.

A mí cuñada Soe por ser tan linda conmigo.

A mi tía Mercedes y mi prima Yuleisy por haberme hecho estos 5 años tan agradables, en general a toda mi familia por darme tanto aliento.

A Izary y Yani por haberme alentado constantemente y por esos momentos tan especiales que son irrepetibles en estos 5 años y a Raidis, Saily y Frances que aunque ya no están aquí también me hicieron muy feliz.

A nuestro co-tutor Abel por su dedicación y orientación.

A nuestra Revolución y nuestro Comandante por darnos todo para poder ser alguien en la vida.

A todos en general, sin que se nos quede nadie…Gracias.

DEDICATORIA

A nuestros padres, quienes nos han brindado amor incondicional y siempre han impulsado a superarnos profesionalmente y ofreciéndonos aliento constante para lograrlo.

A nuestras hermanas, hermanos y familiares en general, que nos apoyan siempre y ayudan a conseguir nuestros anhelos.

RESUMEN

La Ingeniería del Software, definida como el estudio de los principios y metodologías para el desarrollo y mantenimiento de sistemas software, va marcando las pautas de cómo se debe trabajar en el desarrollo de sistemas de información dentro de la ingeniería informática. Sin embargo, el término de la Ingeniería del Software es un término amplio que abarca multitud de sistemas y que engloba un gran número de áreas de investigación. Una de estas áreas es la que se ha denominado *multimedia*. En base a eso gira el presente trabajo.

En el trabajo se investiga sobre las diferentes metodologías de desarrollo de software multimedia, se realiza un minucioso análisis comparativo entre ellas. Y se propone el uso de una de estas metodologías en los proyectos que desarrollan software multimedia en la Universidad, a partir de los resultados satisfactoriamente obtenidos de un proyecto que utiliza la metodología propuesta.

ÍNDICE

INTRODUCCIÓN

Hoy en día las aplicaciones que se desarrollan son muy diferentes a las que se desarrollaban hace unos años. En este sentido, muchas definiciones de sistemas se están dando dentro del mundo informático, tal es el caso de los sistemas multimedia y sistemas hipermedia. Este trabajo es el resultado de la investigación que se llevará a cabo para proponer el uso de la metodológica adecuada para el desarrollo de software multimedia de los proyectos productivos de la Universidad de las Ciencias Informáticas (UCI), específicamente en la facultad 8. En él se hace un recorrido de todas las propuestas que se están aplicando en este ámbito, se hace una comparación entre ellas, destacando sus ventajas y desventajas y se propone cuál de éstas es la más completa para su uso en la UCI.

La **situación problémica** es que en la Universidad de las Ciencias Informáticas, específicamente en la facultad 8 no existe suficiente documentación acerca de las distintas metodologías de desarrollo de software multimedia, por lo que el conocimiento y el uso de éstas no es el adecuado. Tal es el caso de que la mayoría de los proyectos multimedia de la facultad no usan metodología, lo que trae como resultado que no existe la suficiente organización a la hora de realizar este producto informático. No se siguen rigurosos protocolos para la firma de acuerdos de entrega de información entre el cliente y el representante del equipo de desarrollo, lo que trae casos de demora y atenta contra el desarrollo del proyecto.

El **problema a resolver** sería ¿Cómo mejorar el proceso de desarrollo de software multimedia, mediante la propuesta del uso de una metodología?

El **objetivo general** es proponer el uso de una metodología adecuada a los proyectos de la facultad 8 a partir de un estudio comparativo de diferentes propuestas metodológicas existentes, para mejorar el proceso de desarrollo de software multimedia.

Los **objetivos específicos** son:

- Realizar un estudio comparativo de las metodologías de desarrollo de software

multimedia, para llegar al consenso de cual de todas ellas es la que más se adecua a las características específicas de los proyectos de la facultad 8

- Realizar entrevistas a los líderes y desarrolladores de los diferentes proyectos multimedia de la facultad 8
- Realizar entrevistas a personal fuera de la Universidad vinculados al tema de las metodologías para desarrollo de software multimedia

El **objeto de estudio** lo constituye el proceso de desarrollo de software multimedia en los proyectos productivos de la Facultad 8.

El **campo de acción** lo constituyen las distintas metodologías de desarrollo de software multimedia.

La **hipótesis** que se plantea es que si se desarrolla un estudio profundo de las diferentes metodologías de desarrollo de software multimedia, se podrá obtener una propuesta metodológica utilizable en los proyectos productivos de la universidad y lograr con ello mayor eficiencia en la producción de software.

El presente trabajo está compuesto por tres capítulos. En el Capítulo 1 se argumentará la investigación científica utilizada en la realización del presente trabajo de diploma. Una serie de conceptos relacionados con el tema en cuestión, tales como software, ingeniería de software, hipertexto, multimedia e hipermedia, el significado de la herramienta CASE y las distintas definiciones metodologías. Se aborda además sus principios, características, necesidades, objetivos y los principales aspectos a seguir para construir o elegir una metodología. Se refiere además al surgimiento de la multimedia en el mundo y en Cuba, el surgimiento de las metodologías de desarrollo de software y los antecedentes de las metodologías para desarrollo de software multimedia, comenzando por las de corte nacional, como Multimet, Metvisual y Cedisac. Más adelante se recogen las metodologías internacionales comenzando por el modelo HDM, las metodologías RMM, EORM, OOHDM, SOHDM, HFPM, OO/Pattern Approach y por ultimo el Proceso Unificado de Software, con la extensión para multimedia OMMMA-L.

En el Capítulo 2 se hace una caracterización más amplia de las metodologías mencionadas en el Capítulo 1, destacando sus diferentes fases y en cual de estas se desarrollan más.
En el Capítulo 3 se realiza un análisis comparativo de las metodologías descritas. Basándose en los resultados de los objetivos específicos y los métodos utilizados para darle cumplimiento, se sientan las bases para definir la propuesta a utilizar en los proyectos multimedia de la facultad 8.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se expone la metodología de investigación utilizada para el desarrollo del mismo. Además brinda una visión general de los aspectos relacionados con el proceso de desarrollo de software multimedia y los conceptos necesarios para el estudio del mismo. Se enfocan los principales aspectos de las metodologías de desarrollo de software multimedia, principios, características y objetivos. Así como el surgimiento de la multimedia en el mundo y en Cuba, emparejado a la evolución de las metodologías y su uso en el país.

1.1. Metodología de la investigación

Es la actividad de búsqueda que se caracteriza por ser reflexiva, sistemática y metódica; tiene por finalidad obtener conocimientos y solucionar problemas científicos, filosóficos o empírico-técnicos, y se desarrolla mediante un proceso.

La investigación científica es la búsqueda intencionada de conocimientos o de soluciones a problemas de carácter científico; *el método científico* indica el camino que se ha de transitar en esa indagación y las técnicas precisan la manera de recorrerlo.

Las metodologías de la investigación cuentan con distintas caracterizaciones pero en el presente trabajo solo se expondrán las que fueron utilizadas para lograr el objetivo del mismo.

- Según el propósito o finalidades perseguidas:

Investigación aplicada: Este tipo de investigación también recibe el nombre de práctica o empírica. Se caracteriza porque busca la aplicación o utilización de los conocimientos que se adquieren.

Investigación documental: Este tipo de investigación es la que se realiza, como su nombre lo indica, apoyándose en fuentes de carácter documental, esto es, en documentos de cualquier especie. Como subtipos de esta investigación encontramos la investigación bibliográfica, la hemerográfica y la archivística; la primera se basa en la consulta de libros, la segunda en artículos o ensayos de revistas y periódicos, y la tercera en documentos que se encuentran

en los archivos, como cartas, oficios, circulares, expedientes, etcétera.

Investigación de campo: Este tipo de investigación se apoya en informaciones que provienen entre otras, de entrevistas, cuestionarios, encuestas y observaciones. Como es compatible desarrollar este tipo de investigación junto a la investigación de carácter documental, se recomienda que primero se consulten las fuentes de la de carácter documental, a fin de evitar una duplicidad de trabajos.

1.2. Marco teórico conceptual

Concepto de Software

El *software* es una producción inmaterial del cerebro humano y tal vez una de las estructuras más complicadas que la humanidad conoce. De hecho, los expertos en computación aún no entienden del todo cómo funciona, su comportamiento, sus paradojas y sus límites. Básicamente, el software es un plan de funcionamiento para un tipo especial de máquina, una máquina *virtual* o *abstracta*. Una vez escrito mediante algún lenguaje de programación, el software se hace funcionar en ordenadores, que temporalmente se convierten en esa máquina para la que el programa sirve de plan. El software permite poner en relación al ser humano y a la máquina y también a las máquinas entre sí. Sin ese conjunto de instrucciones programadas, los ordenadores serían objetos inertes, como cajas de zapatos, sin capacidad siquiera para mostrar algo en la pantalla.

Concepto de Ingeniería de Software

Consideramos completa la definición de Fritz Bauer quien expresa que "*la Ingeniería de Software* consiste en el establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales".

La Ingeniería de Software está conformada por:

Herramientas: Soporte automático o semiautomático a los métodos, orientadas a etapas particulares en el diseño de un software. Herramientas CASE.

Métodos: Cómo se construye el software (planificación, análisis de los requisitos, diseño del sistema, codificación, prueba y mantenimiento).

Procedimientos: Secuencia en que se aplican los métodos, entregas y controles. Son los que unen los métodos con las herramientas.

El ciclo de vida clásico para el desarrollo de software permanece como el modelo procedimental más ampliamente usado por los ingenieros del software. Sin embargo, con el paso de los años se han producido críticas a este paradigma. Entre los problemas que se presentan se encuentran:

- Las dificultades del cliente para establecer explícitamente al principio todos los requerimientos
- Una versión, funcionando, del programa no estará disponible hasta las etapas finales del desarrollo del proyecto
- Los costos de mantenimiento suelen superar a los costos de desarrollo

Concepto de hipertexto

Hipertexto: Formato que se le aplica a un texto, en el cual se representan palabras claves (en la mayoría de los casos subrayados o con otros colores) las cuales dan acceso a una información determinada. La forma más habitual de hipertexto en documentos es la de hipervínculos o referencias cruzadas automáticas que van a otros documentos.

Concepto de multimedia

Multimedia: Es cualquier combinación de texto, arte gráfico, sonido, animación y vídeo que llega a nosotros por computadora u otros medios electrónicos. Es un tema presentado con lujos de detalles.

Concepto de hipermedia

Hipermedia: Forma de presentación de la información estructurada en nodos. Cada nodo de información puede incluir textos, imágenes, videos, animaciones, gráficos y sonidos. Cualquiera de estos medios puede convertirse en un enlace con otro nodo y el usuario puede acceder a otro nivel de información utilizando no solo el texto.

Concepto de herramientas CASE

CASE (Computer Aided Software Engineering). Es un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información.

Las herramientas CASE abarcan todos los pasos del proceso de software, y también aquellas actividades generales que se aplican a lo largo de todo el proceso. Las herramientas CASE son un complemento de la caja de herramientas del ingeniero del software, además ayudan a asegurar que la calidad sea algo diseñado antes de llegar a construir el producto. (*Fundamentos de ingienieria de software*)

Objetivos:

- Permitir la aplicación práctica de las metodologías respaldadas por la herramienta
- Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones
- Simplificar el mantenimiento de los programas
- Mejorar y estandarizar la documentación
- Aumentar la portabilidad de las aplicaciones
- Facilitar la reutilización de componentes software
- Permitir un desarrollo y un refinamiento visual

Concepto de metodología

Maddison en 1983 define metodología como un conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de sistemas de información. Por lo tanto, una *metodología* es un conjunto de componentes que especifican:

- Cómo dividir un proyecto en etapas.
- Qué tareas se llevarán a cabo en cada etapa.
- Qué salidas se producen y cuando deben producirse.
- Que restricciones se aplican.
- Que herramientas van a ser utilizadas.

- Como se gestiona y controla el proyecto.

De un modo mas general una metodología podría definirse como “un conjunto de conceptos para poder abstraer el dominio del problema, una a notación para representar esos conceptos, una serie de pasos y procedimientos a seguir”.

En un proyecto de desarrollo de software la *metodología* define Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. No existe una metodología de software universal. Las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigen que el proceso sea configurable.

Para desarrollar un proyecto de software es necesario establecer un enfoque disciplinado y sistemático. Las metodologías de desarrollo influyen directamente en el proceso de construcción y se elaboran a partir del marco definido por uno o más ciclos de vida.

EL presente trabajo concuerda lo dicho por Piattini en 1996, que no hay un consenso entre los autores sobre el concepto de metodología, y por lo tanto no existe una definición universalmente aceptada. Sí hay un acuerdo en considerar a la metodología como “un conjunto de pasos y procedimientos que deben seguirse para el desarrollo del software”.

1.3. Principios de las metodologías

Sobre la metodología y su utilización se han establecido algunos principios extraídos de la práctica y que son comunes en el desarrollo de productos informáticos en general, no sólo para las multimedia.

- La metodología es un medio para desarrollar productos informáticos
- La metodología abarca todo el proceso de desarrollo de los productos informáticos
- La metodología establece las etapas, procedimientos y documentos que se han de realizar para desarrollar los productos informáticos
- La metodología proporciona una norma para definir claramente y sin ambigüedad todos los elementos (conceptuales, estructurales y formales) que conforman un producto informático

- La metodología se aplica flexiblemente pero sin excepción al desarrollo de todos los productos informáticos
- La metodología se perfecciona con la práctica

1.4. Características, necesidades y objetivos de las metodologías

Se pueden enumerar una serie de características que debe tener la metodología y que influirán en el entorno de desarrollo:

- Existencia de reglas predefinidas
- Cobertura total del ciclo de desarrollo
- Verificaciones en cada etapa
- Flexibilidad: aplicación en un amplio espectro de casos
- De fácil comprensión
- Soporte de herramientas automatizadas
- Planificación y control
- Comunicación efectiva
- Utilización sobre un abanico amplio de proyectos
- Fácil formación
- Actividades que mejoren el proceso de desarrollo

Las metodologías persiguen tres necesidades principales:

- Mejores aplicaciones, tendientes a una mejor calidad, aunque a veces no es suficiente
- Un proceso de desarrollo controlado, que asegure el uso de los recursos apropiados y un costo adecuado
- Un proceso estándar en la organización, que no sienta los cambios del personal

Las metodologías tienen diferentes objetivos, los cuales son:

- Brindar un método sistemático, de modo de controlar el progreso del desarrollo del proyecto
- Especificar los requerimientos de un software en forma apropiada
- Construir productos bien documentados y de fácil mantenimiento
- Ayudar a identificar las necesidades de cambio lo más pronto posible
- Proporcionar un sistema ágil que satisfaga a todas las personas involucradas
- Planificación y control del proyecto
- Comunicación efectiva entre desarrolladores y usuarios
- Que soporte reusabilidad del software

1.5. Aspectos a seguir para construir o elegir una metodología

En el momento de adoptar un estándar o construir una metodología, se han de considerar unos requisitos deseables, por lo que seguidamente se proponen una serie de criterios de evaluación de dichos requisitos.

La metodología debe ajustarse a los objetivos

Cada aproximación al desarrollo de software está basada en unos objetivos. Por ello la metodología que se elija debe recoger el aspecto filosófico de la aproximación deseada, es decir que los objetivos generales del desarrollo deben estar implementados en la metodología de desarrollo.

La metodología debe cubrir el ciclo entero de desarrollo de software

Para ello la metodología ha de realizar unas etapas:

- Investigación
- Análisis de requisitos
- Diseño

La metodología debe integrar las distintas fases del ciclo de desarrollo

- **Rastreabilidad.** Es importante poder referirse a otras fases de un proyecto y fusionarlo con las fases previas. Es importante poder moverse no sólo hacia adelante en el ciclo de vida, sino hacia atrás de forma que se pueda comprobar el trabajo realizado y se puedan efectuar correcciones.
- **Fácil interacción entre etapas del ciclo de desarrollo.** Es necesaria una validación formal de cada fase antes de pasar a la siguiente. La información que se pierde en una fase determinada queda perdida para siempre, con un impacto en el sistema resultante.

La metodología debe incluir la realización de validaciones

La metodología debe detectar y corregir los errores cuanto antes. Uno de los problemas más frecuentes y costosos es el aplazamiento de la detección y corrección de problemas en las etapas finales del proyecto. Cuanto más tarde sea detectado el error más caro será corregirlo. Por lo tanto cada fase del proceso de desarrollo de software deberá incluir una actividad de validación explícita.

La metodología debe soportar la determinación de la exactitud del sistema a través del ciclo de desarrollo

La exactitud del sistema implica muchos asuntos, incluyendo la correspondencia entre el sistema y sus especificaciones, así como que el sistema cumple con las necesidades del usuario. Por ejemplo, los métodos usados para análisis y especificación del sistema deberían colaborar a terminar con el problema del entendimiento entre los informáticos, los usuarios, y otras partes implicadas. Esto implica una comunicación entre usuario y técnico amigable y sencillo, exento de consideraciones técnicas.

La metodología debe ser la base de una comunicación efectiva

Debe ser posible gestionar a los informáticos, y éstos deben ser capaces de trabajar conjuntamente. Ha de haber una comunicación efectiva entre analistas, programadores,

usuarios y gestores, con pasos bien definidos para realizar progresos visibles durante la actividad del desarrollo.

La metodología debe funcionar en un entorno dinámico orientado al usuario

A lo largo de todo el ciclo de vida del desarrollo se debe producir una transferencia de conocimientos hacia el usuario. La clave del éxito es que todas las partes implicadas han de intercambiar información libremente. La participación del usuario es de importancia vital debido a que sus necesidades evolucionan constantemente. Por otra parte la adquisición de conocimientos del usuario le permitirá la toma de decisiones correctas.

Para involucrar al usuario en el análisis, diseño y administración de datos, es aconsejable el empleo de técnicas estructuradas lo más sencillas posible. Para esto, es esencial contar una buena técnica de diagramación.

La metodología debe especificar claramente los responsables de resultados

Debe especificar claramente quienes son los participantes de cada tarea a desarrollar, debe detallar de una manera clara los resultados de los que serán responsables.

La metodología debe poder emplearse en un entorno amplio de proyectos software

- *Variedad.* Una empresa deberá adoptar una metodología que sea útil para un gran número de sistemas que vaya a construir. Por esta razón no es práctico adoptar varias metodologías en una misma empresa.
- *Tamaño, vida.* Las metodologías deberán ser capaces de abordar sistemas de distintos tamaños y rangos de vida.
- *Complejidad.* La metodología debe servir para sistemas de distinta complejidad, es decir puede abarcar un departamento, varios de departamentos o varias empresas.
- *Entorno.* La metodología debe servir con independencia de la tecnología disponible en la empresa.

La metodología se debe poder enseñar

Incluso en una organización sencilla, serán muchas las personas que la van a utilizar, incluso los que se incorporen posteriormente a la empresa. Cada persona debe entender las técnicas

específicas de la metodología, los procedimientos organizativos y de gestión que la hacen efectiva, las herramientas automatizadas que soportan la metodología y las motivaciones que subyacen en ella.

La metodología debe estar soportada por herramientas CASE

La metodología debe estar soportada por herramientas automatizadas que mejoren la productividad, tanto del ingeniero de software en particular, como la del desarrollo en general. El uso de estas herramientas reduce el número de personas requeridas y la sobrecarga de comunicación, además de ayudar a producir especificaciones y diseños con menos errores, más fáciles de probar, modificar y usar.

La metodología debe soportar la eventual evolución del sistema

Normalmente durante su tiempo de vida los sistemas tienen muchas versiones, pudiendo durar incluso más de 10 años. Existen herramientas CASE para la gestión de la configuración y otras denominadas "Ingeniería inversa" para ayudar en el mantenimiento de los sistemas no estructurados, permitiendo estructurar los componentes de éstos facilitando así su mantenimiento.

La metodología debe contener actividades conducentes a mejorar el proceso de desarrollo de software

Para mejorar el proceso es básico disponer de datos numéricos que evidencian la efectividad de la aplicación del proceso con respecto a cualquier producto software resultante del proceso. Para disponer de estos datos, la metodología debe contener un conjunto de mediciones de proceso para identificar la calidad y coste asociado a cada etapa del proceso. Sería ideal el uso de herramientas CASE.

1.6. Surgimiento de la multimedia en el mundo y en Cuba

La multimedia surge en el mundo con el desarrollo del software gráfico (Apple Macintosh, 1984; WINDOWS 2.0, 3.0 y 95, en 1986, 1990 y 1995, respectivamente), y el hardware potente y veloz. Inmediatamente se le vio las inmensas posibilidades comunicativas que tenía

para el turismo, la educación, la ciencia y los negocios. Muchas instituciones cubanas asimilaron las nuevas tecnologías y se empeñaron en producir diferentes títulos multimedia.

Entre los años 1988 y 1992 varias instituciones empiezan a interesarse por el tema de las multimedias, una de ellas fue la Academia de Ciencias con el Sistema de Información Geográfico que hacia uso de imágenes y elementos gráficos visuales.

Uno de los primeros laboratorios de multimedia que hubo en el país en el año 1992 fue el establecido con ayuda de la UNESCO en la Facultad de Economía de la Universidad de La Habana, el cual quedó pronto superado por los de otras instituciones que se han especializado en la temática.

En 1992 CEDISAC, actual CITMATEL, hace sus primeros productos multimedia en disketes llamado Análisis Estructurado. Entre el 1992 y el 1994 comienzan a prepararse para una edición en CD, que se llamó NATURA, se presentó en el evento Internacional Informática ’94 y fue la primera producción sobre CD ROM cubano, no incluía música.

En 1995 CESOFT, que actualmente no existe, junto con la empresa colombiana Kimera y en unión con el MINED producen NATURA como si fuera la primera obra multimedia cubana, porque incluía música.

En ese mismo año INFOMASTER, perteneciente al Ministerio de Educación Superior (MES) da origen a WITS (Software para aprender Windows).

Entre el 1994 y el 1996 CEISIC del Ministerio de Cultura, se produce una salida masiva de multimedias cubanas de CD ROM.

AL final de la década del ’90 ya comienzan a producirse excelentes productos multimedia en campos como la medicina, el arte, la música, la educación la geografía, la historia el comercio y otros.

1.7. Surgimiento de las metodologías de desarrollo de software

Desde que el desarrollo de aplicaciones informáticas se empezó a considerar un proceso de ingeniería, muchas metodologías de desarrollo han ido naciendo con el fin de dar soporte al

ciclo de desarrollo del proyecto. Entre estas, podemos destacar algunas como MERISE elaborada en 1978 (Metodología de Análisis y Diseño de Sistemas de Información, define un conjunto de etapas: estudio preliminar, estudio detallado, implementación, realización y puesta en marcha), SSADM creada en 1981 (Structures Systems Analysis and Design Method, proporciona un conjunto de procedimientos para llevar a cabo el análisis y diseño, pero no cubre aspectos como la planificación estratégica ni entra en la construcción del código), MÉTRICA creada en 1989 (Metodología que constituye una guía sencilla y útil para la Planificación, Análisis, Diseño, Construcción e Implantación de Sistemas de Información.) y ya más recientes como OMT en 1991 (Object Modelling Technique), o el actual UML.

Todos estas metodologías estaban orientadas desde sus comienzos al desarrollo de sistemas para gestionar una información que se encuentra almacenada en una o varias bases de datos, distribuidas o no. Entre los aspectos que se tratan como más críticos en estas metodologías, los más importantes son el almacenamiento y la recuperación adecuada de la información, así como que las posibilidades funcionales que ofrezcan sean las necesarias.

1.7.1. Antecedentes de las metodologías de desarrollo de software multimedia

Otro término que comienza a nacer es el de las aplicaciones multimedia. Mientras que las metodologías anteriores tratan con especial interés los aspectos de almacenamiento y funcionalidad, en las aplicaciones multimedia el objetivo esencial va a ser difundir información almacenada en diferentes medios (imágenes, vídeos, música) de manera que llegue al público no experto en informática de una forma sencilla, fácil e intuitiva.

El desarrollo de estos sistemas comienza a realizarse sin que haya ninguna norma que se pueda tomar como marco de referencia para su desarrollo. Sin embargo, a principios de los 90, se comienza a estudiar la necesidad de una metodología que guíe a los desarrolladores y que asegure la calidad de los productos multimedia generados. Por esta razón, desde el año 93 comienzan a publicarse propuestas metodológicas y nuevos modelos para representar la

problemática de estas aplicaciones. En este caso han ayudado también metodologías cubanas, las cuales recogen lo mejor de la tecnología mundial.

En Cuba, es conocida la metodología **MultiMet** para modelar el proceso de creación de una multimedia, orientada a las etapas de concepción más que a la descripción de la modelación del producto como tal. Aunque es capaz de guiar las acciones circundantes a la fabricación, deja un hueco en el conocimiento necesario para la estructura programática del software y el flujo de procesos durante el mismo, así como instrumentos que faciliten el análisis, diseño e implementación.
Además también son conocidas en nuestro país la metodología **MetVisual** para cuando que se desarrolle un proyecto o aplicación informática haciendo uso de las técnicas estructuradas y los entornos o medios visuales de programación como Microsoft Visual Basic, Microsoft Acces, Microsoft Visual Foxpro y otros similares que se basan en técnicas estructuradas, esta metodología es recomendable para la construcción de sistemas fuertemente interactivos. A diferencia de la forma de trabajo tradicional de otras metodologías estructuradas donde primero se realiza el análisis, después el diseño y por ultimo el desarrollo, en esta metodología estas etapas se realizan de forma concurrente por lo que las etapas de la metodología no coinciden con las establecidas en otras metodologías. Y la metodología **CEDISAC** (Centro de Diseño de Sistemas Automatizados) desarrollada y utilizada en la empresa CITMATEL por el MSc. Rafael Barrera Yanes; obtenida a partir de la experiencia acumulada durante varios años (1992-1996). Constituye un proceso para construir una obra de comunicación audiovisual interactivo, basada en la praxis obtenida.

El modelo HDM (Hypertext Design Model) (FRANCA GARZOTO 1993) es uno de los primeros métodos desarrollado para definir la estructura y la navegación propia de las aplicaciones multimedia. HDM se basa en el modelo Entidad-Relación, aunque amplía el concepto de entidad e introduce nuevos elementos, como las unidades o los enlaces.

HDM no supone una metodología para el desarrollo de aplicaciones multimedia, es simplemente una técnica de modelado. Es cierto que los elementos definidos por HDM sirven para definir este tipo de aplicaciones, pero resultan insuficientes para guiar al diseñador en el proceso de desarrollo de las mismas. HDM va a sentar las bases para futuras propuestas de desarrollo, ofreciendo ideas como la separación de lo conceptual, información que se almacena, y de la presentación, información que se presenta.

RMM (RelatioShip Management Methodology) (TOMAS IZAKOWITZ 1995) es propuesta en 1995 por Tomas Izsakowitz, Arnold Kamis y Marios Kounfaris. Esta metodología se define como un proceso de análisis, diseño y desarrollo de aplicaciones multimedia. Los elementos principales de RMM son el modelo E-R (Entidad-Relación) y el modelo RMDM (Relationship Management Data Model) basado en el modelo HDM. El modelo RMDM propone un lenguaje que permite describir los objetos del dominio, sus interrelaciones y los mecanismos de navegación hipermedia de la aplicación.
Esta metodología es apropiada para dominios con estructuras regulares (es decir, con clases de objetos bien definidas, y con claras relaciones entre esas clases). Por ejemplo, catálogos o "frentes" de bases de datos tradicionales. Según sus autores, está orientada a problemas con datos dinámicos que cambian con mucha frecuencia, más que a entornos estáticos.

EORM (Enhanced Object Relationship Methodology) es una de las metodologías de diseño de aplicaciones multimedia más referenciadas. Nace a partir de RMM y HDM pero se orienta ya al paradigma Orientado a Objetos. EORM propone un proceso iterativo que consiste en enriquecer un modelo de objetos para representar las relaciones existentes entre objetos (enlaces). EORM también es adecuada porque, siguiendo la idea inicial de HDM, separa la navegación de lo conceptual. Esto garantiza la reutilización y un mantenimiento más fácil. Si hay un cambio en la navegación, lo conceptual no se modifica.

OOHDM (Object Oriented Hypermedia Design Model) (GUSTAVO ROSSI 1997) metodología propuesta por Rossi y Schwabe está basada en HDM. OOHDM es una propuesta basada en el diseño, que ofrece una serie de ideas que han sido asumidas por bastantes propuestas y que han dado muy buenos resultados. La primera de ellas es que hace una separación clara entre lo conceptual, lo navegacional y lo visual. Esta independencia hace que el mantenimiento de la aplicación sea mucho más sencillo. Además, es la primera propuesta que hace un estudio profundo de los aspectos de interfaz, esencial no solo en las aplicaciones multimedia, sino que es un punto crítico en cualquiera de los sistemas que se desarrollan actualmente.
OOHDM es también orientada a objetos, hace uso de un diagrama tan estandarizado como el de clases, para representar el aspecto de la navegación a través de las clases navegacionales: índices, enlaces y nodos. Esta idea ha dado muy buenos resultados y parece muy adecuada a la hora de trabajar.

SOHDM (Scenario-based Objecto-oriented Hypermedia Design Methodology) (HEESEOK LEE 1998) es una metodología realizada por Heeseok Lee, Choongseok Lee, Cheonsoo Yoo SOHDM. Es hasta ahora la única propuesta que tiene en cuenta aspectos como la especificación de requisitos haciendo uso de los escenarios. Es una propuesta bastante interesante pues cubre todas las fases del proceso de desarrollo, obviando la implantación y las pruebas.

HFPM (Hypermedia Flexible Process Modeling Strategy) (OLSINA 1998) fue propuesta por Luis Olsina en 1998. Esta metodología engloba todas las fases del proceso de desarrollo, va desde el análisis hasta el desarrollo de la documentación y el mantenimiento. Además divide y detalla cada una de las tareas que comprende cada fase.

OO/Pattern Approach (J. THOMSON 1998) fue propuesta por Thomson, Greer y Cooke en 1998. Esta propuesta es bastante similar a HFPM pues ambas proponen el uso de patrones y

es orientada a objetos para el diseño navegacional y la interfaz. Sin embargo, esta propuesta, a diferencia de HFPM, no cubre el ciclo completo de desarrollo.

El Proceso Unificado de Software (RUP) es un proceso desarrollado por Philippe Kruchten, Ivar Jacobson y otros de la Corporación “Rational Software”, ahora una división de IBM, cuyo objetivo es producir software de alta calidad, es decir, que cumpla con los requerimientos de los usuarios dentro de una planificación y presupuesto establecidos. Aunque RUP abarca un determinado número de actividades diferentes, está diseñado para poder ajustarse en la selección de procesos específicos destinados a un proyecto u organización de desarrollo en particular y es reconocida en medio de grandes equipos de trabajo que llevan a cabo el manejo de complicadas aplicaciones de software.

RUP toma en cuenta las mejores prácticas en el modelo de desarrollo de software en particular las siguientes:

- Desarrollo de software en forma iterativa
- Manejo de requerimientos
- Utiliza arquitectura basada en componentes
- Modela el software visualmente (Modela con Unified Modeling Language, UML)
- Verifica la calidad del software
- Controla los cambios

El ciclo de vida del Proceso Unificado es en la práctica un ciclo de vida en espiral. Sus características esenciales es que se trata de un ciclo de vida incremental e iterativa. Iterativo porque se producen varios ciclos en su desarrollo en cada uno de los cuales se concreta más el producto resultado del ciclo anterior. E incremental porque en cada ciclo, el producto resultante se va a adecuar más a las necesidades de los clientes.

RUP se aplica a una buena cantidad de productos y procesos de software en el mundo. No es específico para diseño hipermedia, sin embargo a través de la extensión de UML para multimedia, conocida por **OMMMA – L**, se presenta como algo eficientemente realizable.

Desarrollada por Stefan Sauer y Gregor Engels, profesores e investigadores del Departamento de Matemática y Ciencias de la Computación de la Universidad de Paderborn, Alemania, **OMMMA-L** integra el comportamiento interactivo con el de procedimientos temporales para lograr la descripción de aplicaciones que reaccionan ante eventos externos y producen ejecuciones dinámicas predecibles en tiempo de ejecución, dando una muestra sólida de la integración temporal y la sincronización de diferentes objetos de media.

Representable a través de los modelos y artefactos, conservando la semántica de muchos de estos y creando nuevas interpretaciones afines a una especificación multimedia, OMMMA-L modela diversos aspectos de sistema basados en el paradigma Orientado a Objeto, utiliza el Lenguaje de Modelado Unificado y se integra dentro del Proceso Unificado de Ingeniería del Software.

Conclusiones

En el capitulo se definieron una serie de conceptos importantes relacionados con el tema tratado, los cuales servirán de soporte al lector para adentrarse en el tema en cuestión. Se abordaron los principios, características, necesidades y objetivos de las metodologías, así como los principales aspectos a seguir para elegir la más adecuada. Se refirió además al surgimiento de la multimedia en el mundo y en Cuba, el surgimiento de las metodologías de desarrollo de software y los antecedentes de las metodologías para desarrollo de software multimedia, comenzando por las de corte nacional y las elaboradas en el resto del mundo.

CAPITULO 2. DESCRIPCIÓN DE LAS METODOLOGÍAS PARA EL DESARROLLO DE SOFTWARE MULTIMEDIA

Introducción

En el presente capítulo se pasará a analizar las metodologías de corte nacional e internacional mencionadas en el capitulo anterior, detallando las fases que desarrollan y en cuál de éstas emplean más esfuerzo.

2.1. Multimet

Las etapas están bien delimitadas y su objetivo es que cada especialista componente del equipo de desarrollo en cada proyecto, conozca la aplicación de forma integral y pueda dirigir su trabajo hacia un fin común.

2.1.1. Etapas de Multimet

Etapa 1- Estudio preliminar

- **Definición del producto**

En este punto deben quedar definidos algunos elementos básicos relacionados con las necesidades de los usuarios, elementos necesarios para el desarrollo y para la ejecución del producto.

- **Elaboración del plan de desarrollo**

Confección de un plan que incluya todas las etapas del desarrollo con fecha de inicio y de terminación y responsable. Puede ser que hasta este momento no se tengan todos los elementos para definir una etapa en particular, en este caso se va completando a medida que se avance.

En este momento se debe precisar el personal necesario para llevar a cabo el proceso, con cual se cuenta, cual debe ser contratado y cuales servicios se solicitarán, es el momento de definir el grupo multidisciplinario que acometerá el trabajo.

- **Estudio de factibilidad**

Para hacer el estudio de factibilidad debemos tener en cuenta dos elementos:

1. La factibilidad económica.
2. La factibilidad técnica

Factibilidad económica:

En este punto deben analizarse varios factores, uno de los más importantes es la relación costos - beneficios, el impacto del producto final, costo de los elementos que hacen falta para el desarrollo y crecimiento potencial en el mercado.

El análisis de costos - beneficios debe tener en cuenta los beneficios tanto económicos como sociales que tendrá el producto, aunque en todos los casos se debe tratar de recuperar la inversión en que se incurra.

En los costos se debe incluir la inversión en la adquisición de equipamiento que en el caso de las aplicaciones Multimedia suelen ser necesarios, además el costos de los servicios técnicos y de la mano de obra que sea necesario contratar. Además incluir los costos en la producción de software donde se tienen en cuenta la digitalización de imágenes, textos, videos, sonido, desarrollo de animaciones, etc.

Factibilidad técnica:

Lo más importante a considerar es:

- Si es posible disponer de todo el personal técnico.

- Si se dispone de la tecnología necesaria, tanto desde el punto de vista de hardware como de software.

Como resultado debe quedar claro si es factible o no desarrollar el producto y continuar con el resto de las etapas.

Etapa 2- Definición del contenido de la aplicación

- **Definición de los objetivos de la aplicación**

En este caso se definen los objetivos desde el punto de vista de la aplicación propiamente dicho, teniendo en cuenta si es educativa, demostrativa, informativa, etc.

- **Identificación de la audiencia**

Uno de los aspectos más importantes es la correcta identificación del usuario final del sistema que puede resolverse respondiendo a la pregunta, *¿a quién va dirigida la aplicación?* , se debe tener en cuenta que los criterios de diseño están en función de satisfacerlos y un correcto análisis en este aspecto permitirá el cumplimiento de los objetivos antes señalados y definir que contenido incluir y como hacerlo. Una audiencia heterogénea puede conducir a que la aplicación tenga que ser estructurada en varios niveles.

- **Especificación del contenido**

Aquí deben destacarse los temas que serán tratados, en su orden de aparición y teniendo en cuenta para cada uno el nivel de detalle y la forma en que será estructurado.

- **Definición de los medios y sus objetivos**

Para cada tema o parte de él debe tratar de definirse con que medios van a ser representados, y para cada medio utilizado debe quedar claro con que objetivo aparecerá, todo esto determina la importancia que tiene su presencia en la aplicación. Los objetivos que pueden señalarse son: educar, informar, persuadir, entretener, complementar o una mezcla de ellos, también hace falta conocer la disponibilidad de cada medio y la fuente de obtención de cada uno. Para esto proponemos utilizar la siguiente tabla:

Tema	Medio	Objetivo	Disponible	Fuente
	• • •			

Se considera disponible cuando se cuenta con el texto, la imagen, el sonido, el vídeo, en cualquiera de los medios de almacenamiento comunes para cada uno de ellos, en caso de que deba crearse, no estará disponible y en ese caso la fuente será el medio para obtenerlo.

- **Establecer normas de diseño**

Para cada medio debe quedar claro que forma tendrá dentro de la aplicación, esto es lo que garantiza la uniformidad. Estas características o parámetros son específicos para cada medio utilizado:

Textos:

- Porciento máximo de ocupación de pantallas.
- Fuentes utilizadas para títulos.
- Fuentes utilizadas para texto normal.

Imágenes: Los parámetros de cada una están muy ligados con los objetivos que tienen en la aplicación, pero deben quedar claras las normas generales sobre todo teniendo en cuenta el espacio en disco de que se dispone, deben fijarse:

- Tamaño máximo y mínimo.
- Profundidad del color.
- Resolución de la imagen.

Sonido: Pueden ser utilizados diferentes tipos de sonido como: música de fondo, locución, efectos, cada uno debe ser tratado por sus características e importancia definiendo:

- Frecuencia de muestreo.
- Precisión del valor de cada muestra.

Vídeo / Animación: Ocupan un importante volumen de disco, luego para la definición de sus características se tendrán en cuenta sus objetivos. No deben incluirse videos de muy larga duración porque esto influye en la capacidad y en la calidad. Los parámetros a fijar son:

- Duración.
- Parámetros de cada imagen.

- Parámetros del sonido.
- Cantidad de cuadros por segundo.

Etapa 3 - Especificación del contenido de la aplicación

- **Recopilación y preparación de los medios**

De acuerdo con las fuentes definidas anteriormente para obtener los medios, se procede a recopilar cada uno de ellos y luego a su preparación que en cada uno tendrá características especiales:

Textos: El texto puede ser almacenado en caracteres o como imagen.

Imágenes: La preparación de las imágenes requiere un nivel de especialización, por los conocimientos de diseño que requiere y las facilidades en el manejo de herramientas especializadas.

Sonido: El sonido puede ser recuperado de disco o estar almacenado en un medio externo a la computadora como cinta CD musical, una locución, etc.

Videos / Animación: La fuente de los videos generalmente es una cinta, por lo tanto deben digitalizarse, para ello debe tener una máquina con tarjeta digitalizadora de vídeo y un software apropiado para ello, posteriormente se pasa al proceso de edición.

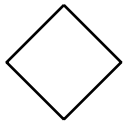
De igual forma para crear las animaciones deben tenerse en cuenta las normas de diseño.

- **Elaboración del diagrama de flujo**

Este diagrama se propone sea igual al de cualquier programa, los símbolos a utilizar son:


Pantalla principal de información


Pantalla de información complementaria


Bifurcación

Navegación

Parece acertada la idea mostrada por otros autores de que los diseñadores puedan definir otros símbolos siempre que sean bien explicados.

- **Confección del guión**

Con el diagrama se obtiene una idea del funcionamiento general e integral del sistema, sin embargo existen aspectos a tener en cuenta y que no se colocan en el diagrama como son:

- Tema tratado en cada elemento.
- Información que aparece en la pantalla.
- Acciones del usuario que determinan las respuestas del sistema.
- Respuesta del sistema a cada acción.
- Tratamiento de errores.

Etapa 4 - Desarrollo de la aplicación

Cuando se llega a este punto en el desarrollo ya está preparada toda la información a incluir y diseñado el funcionamiento integral del sistema desde el punto de vista de las acciones del usuario, queda lo relacionado con la integración de todos los medios a partir de una prueba exitosa del guión y el diagrama de flujo.

- **Comprobación del Diagrama de flujo y acciones de acuerdo al guión**

Debe existir total correspondencia entre lo que ilustra el diagrama de flujo y lo que aparece en el guión, esto debe ser revisado cuidadosamente teniendo en cuenta que la secuencia Acción-Respuesta par cada elemento del diagrama tenga sentido en todos los casos y esté correctamente expresado en el guión.

- **Selección del lenguaje de programación o sistema de autor**

Aquí se trata de seleccionar la herramienta de ensamblaje de la aplicación que debe cumplir algunos requisitos como son:

- Programación visual para garantizar eficiencia.
- Facilidades para la manipulación de recursos Multimedia.

Estas herramientas pueden ser:

- Lenguaje de programación: Lenguajes que permiten la programación visual de aplicaciones, incidiéndose en el uso de bibliotecas de control de dispositivos.
- Sistemas de Autor: Pueden ser de diferentes tipos:

Basados en Líneas de Tiempo: Estas herramientas se vuelven cada vez más comunes dentro de los programas de autoría. Cada una de ellas usa su propia interfaz de usuario para manejar los eventos en el tiempo; muchas utilizan una línea de tiempo para darle secuencia a los eventos que suceden durante una presentación multimedia y con frecuencia despliegan niveles de elementos de esta, ó eventos a lo largo de una escala con incrementos altamente precisos; otros organizan largas secuencias de cuadros gráficos y adicionan el factor de tiempo ajustando cuadro a cuadro la ejecución de la aplicación

Basados en diagramas de flujo: La lógica de la aplicación se estructura a manera de un diagrama de flujo lógico, en el cual se visualizan las trayectorias y los elementos que forman parte de ellas. Permiten el control directo de la interacción con el usuario, identificando sus respuestas y definiendo líneas de acción ante cada una de ellas.

Basados en páginas: La aplicación se estructura en páginas en las cuales existen tanto criterios de linealidad como rutas alternativas, definidas en función de acciones del usuario. Permiten el control de estas acciones.

- **Integración del contenido y los medios en su forma final**

Una vez expresados todos los elementos del diseño y seleccionado el lenguaje de programación, se ensamblan todos los elementos desarrollando un producto de software, esta tarea es responsabilidad de los programadores que deben ajustarse al guión y utilizar las normas de diseño definidas y las bibliotecas con la información. Debe tenerse en cuenta por parte de los especialistas la confección de la Ayuda del sistema.

Etapa 5 - Pruebas de la aplicación

En ningún proyecto informático debe pasarse por alto el proceso de pruebas que es el que garantiza la salida de un producto de calidad. Un software Multimedia debe revisarse desde dos puntos de vista:

- Solidez de la información
- Adecuado funcionamiento

Solidez de la información: Toda la información contenida en la aplicación debe ser verificada en cuanto a:

- No existencia de errores ortográficos
- Calidad de los medios que se muestran
- Correspondencia entre el tema tratado, el texto y el resto de los medios que aparecen en cada pantalla
- Cumplimiento de las normas de diseño

Adecuado funcionamiento: En este caso se trata de comprobar que cada acción del usuario tenga una respuesta correcta del sistema y que no ocurran errores imprevistos.

- **Elaborar protocolo de pruebas**

Cada uno de los puntos de vista antes expresado puede ser revisado a la vez o por separado, esto depende de la complejidad del sistema, el tamaño de la aplicación o la conveniencia a la hora de efectuar las pruebas.

- Solidez de la información: Revisión de cada pantalla verificando los tipos de errores de acuerdo a lo ya tratado y comprobando con las fuentes y las normas de diseño establecidas.
- Adecuado funcionamiento: De acuerdo con el diagrama de flujo y el guión debe elaborarse un protocolo de pruebas que garantice el recorrido a todas las vías posibles y que además permita comprobar si el programa se ajusta completamente a lo deseado.

- **Revisión y comprobación por el usuario**

Crear un grupo para realizar las pruebas, ajeno a los realizadores del producto que las ejecutarán de acuerdo al calendario fijado y guiados por el protocolo.

Etapa 6- Preparación para su distribución

Un producto que utiliza técnicas de Multimedia es en muchos casos un gancho en el mercado, por lo que si se decidió su comercialización debe prestársele la máxima atención.

- **Determinación de la forma de distribución**

Debe decidirse si se distribuirá utilizando disquete o CD-ROM, la tendencia actual es a distribuir los productos en CD-ROM como ya se ha planteado, pero pueden aparecer causas que determinen utilizar los disquetes, como por ejemplo: la no tenencia de lectores entre los usuarios potenciales, no ocupar gran capacidad de memoria la aplicación que no justifica los gastos y el trabajo si tenemos en cuenta que en Cuba no se hace la producción de CD.

- **Diseño de la empaquetadura**

Se entiende como empaquetadura, el medio que se utilizará para contener los discos que componen el producto.

Debe incluirse:

- Para disquetes: Diseño de la etiqueta
- Para CD: Diseño de carátula.

El tipo de empaquetadura (cajas plásticas, sobres de nylon, cartulina, etc.) se selecciona teniendo en cuenta la disponibilidad en el mercado, el tipo de producto, el presupuesto de que se dispone. Cualquiera sea el tipo seleccionado deben diseñarse los materiales asociados a la misma y una portada.

- **Preparación para su producción**
  - Si son disquetes: Preparación de instalador.
  - Preparación del primer juego de discos.
  - Prueba de la aplicación preparada desde discos.
  - Envío a producción.

- Si es CD-ROM: Preparación de instalador
- Preparación en disco duro de una simulación del contenido del CD.
- Quema del premaster.
- Prueba de la aplicación desde el premaster.
- Envío a producción.

- **Elaboración de documentos comerciales**

  - Si se decidió desde un inicio producir una aplicación para su comercialización, debe trabajarse y en paralelo con el resto de las etapas en las líneas y estrategias de comercialización con el estudio de todos los clientes potenciales y la preparación de todos los materiales que permitirán promocionar el producto. Si no se cuenta con un Departamento comercial que pueda desarrolla con éxito la tarea, ésta debe ser confiada a Empresas especializadas.

Como resultado debe quedar:

- Producto final empaquetado
- Documentos comerciales

2.2. CEDISAC

Esta metodología pretende estructurar y describir con cierta formalización el desarrollo de productos informáticos multimedia. Es obtenida a partir de la experiencia acumulada durante varios años por los técnicos y profesionales de la División de Multimedia y Nuevas Tecnologías del Centro de Diseño de Sistemas Automatizados (CEDISAC) del Ministerio de Ciencia, Tecnología y Medio Ambiente (CITMA).

2.2.1. Etapas de CEDISAC

Etapa 1- Concepción del proyecto

En esta etapa se definen los objetivos del proyecto, se identifica el mercado y usuario final, se precisan las fuentes informativas que integran el producto, se perfila la estructura general que

tendrá el producto, se elaboran muestras computacionales de carácter promocional y técnico-funcional y se evalúa la factibilidad técnico-económica para acometer el proyecto.

Etapa 2- Elaboración del guión

Esta etapa es decisiva para lograr un producto realmente competitivo y que reúna las condiciones para clasificarlo como un producto multimedia atractivo. El guión del producto debe contemplar el diseño de la base informativa, la estructura orgánica y funcional que tendrá el producto y la forma en que se presentará la información. La participación de los diseñadores gráficos es crucial en esta etapa, por lo que deben colaborar estrechamente con los analistas para lograr un diseño acorde con los objetivos del proyecto.

Etapa 3- Preparación de la Base informativa.

Es la etapa más ardua y extensa, pues deben recopilarse y procesarse los diversos y enormes volúmenes de información que por lo general integran el producto multimedia. Forman parte de esta etapa la elaboración de las animaciones, siendo usualmente la tarea más difícil de realizar.

Etapa 4- Ejecución del proyecto.

En esta etapa se requiere un cuidado especial. El mejor diseño concebido en la etapa II puede quedar echado por tierra si no se es exigente con el desarrollo de esta etapa. Es la más compleja y en ella se da solución técnica apropiada a lo exigido por el guión. La confección de los hipertextos, la programación y el montaje de todo el producto, así como la confección de la documentación y la ayuda, son las tareas básicas que integran la etapa.

Etapa 5- Revisión y depuración.

Durante todo la producción, es decir, desde la primera etapa, es necesario contar con un sistema de control de la calidad que garantice los resultados esperados, pero una vez

concluido el producto es imprescindible realizar una inspección final en la que debe hacerse una revisión minuciosa, exhaustiva y con el máximo rigor de toda la información, tanto textual como gráfica, de audio, de video y de las animaciones. También se verificará la forma en que se opera el producto y las funciones que debe realizar. Se pulirán los elementos formales de presentación de la información y el diseño de pantallas. Esta etapa es de obligado cumplimiento, máxime si el producto será soportado sobre CD-ROM.

Etapa 6- Premasterización para CD-ROM.

Esta etapa, con la que culmina el desarrollo de un producto multimedia clásico, no habrá que realizarla si el producto no será distribuido sobre disco compacto, pero de ser así hay que comenzar por hacer una grabación de respaldo sobre cinta magnética u otro soporte que así lo permita. De esta forma se puede llevar el producto al disco duro de la microcomputadora con la que se hará la grabación del premáster, previa conversión al formato especificado por la norma ISO 9660 mediante un programa diseñado especialmente para ello. Finalmente se graba un CD-R u otro soporte apropiado que constituye el premáster listo para el proceso industrial que hará las copias de los CD-ROM con el producto multimedia elaborado.

2.3. HDM (Hypermedia Design Model)

HDM propone un conjunto de elementos que permiten al diseñador especificar una aplicación. Estos elementos son las entidades, los componentes, las perspectivas, las unidades y los enlaces. Todos estos elementos pueden incorporarse en la semántica del clásico modelo Entidad-Relación. Sin embargo, y a pesar de que términos como las entidades hayan sido heredados de los ERD (Diagrama entidad relación) a continuación se verá que han sido extendidos para poder representar una estructura compleja que contenga enlaces y una semántica de navegación interna.

En definitiva una aplicación especificada mediante un modelo HDM consiste en una estructura general compuesta por unas unidades básicas denominadas entidades. Una

entidad denota un objeto físico o conceptual del universo de discurso de la aplicación. En HDM las entidades son agrupadas en tipos de entidad. Los tipos de entidad se caracterizan por un nombre, por un conjunto de perspectivas bajo las que se pueden presentar su contenido y un conjunto de enlaces de aplicación por los que se puede navegar.

Una entidad es la unidad mínima autónoma de cualquier modelo HDM, pero existen otros conceptos añadidos que veremos a continuación.

Cada entidad está compuesta por una jerarquía de componentes que heredan las propiedades de dicha entidad. Los componentes no tienen razón de ser sin que exista la entidad de la que dependen. Los componentes son, por su parte, abstracciones para diseñar un conjunto de unidades o nodos que representan un mismo conjunto de información de la entidad. Una unidad, es pues un depósito de la información contenida en una aplicación. Una unidad representa un fragmento del contenido de una entidad presentada bajo una perspectiva particular.

De esta forma, la perspectiva permite representar la multiplicidad de presentaciones de un mismo contenido de información (por ejemplo, la presentación de un documento en múltiples lenguas). Para entender mejor la relación entre estos elementos, se usará la potencia representativa de UML (IVAR JACOBSON 1999). En la figura número 1 se muestra un modelo de clases que representa las relaciones entre los elementos de HDM.

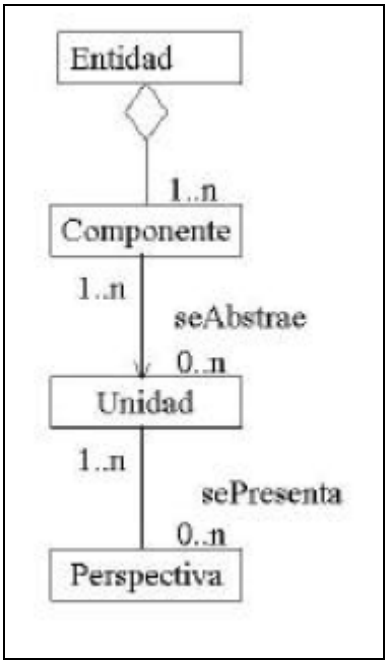


Figura 1. Relaciones de los elementos de HDM

En un modelo HDM las estructuras de información pueden ser conectadas mediante enlaces. Un enlace entre dos elementos indica que en la aplicación hipermedia resultante existe la posibilidad de navegar entre esos dos elementos. HDM distingue tres tipos de enlaces:

- Enlaces estructurales, conectan componentes de la misma entidad.
- Enlaces de perspectiva, conectan perspectivas que corresponden a una misma unidad.
- Enlaces de aplicación, sirven para conectar componentes y unidades. Son los más complejos de los tres pues pueden conectar elementos de unidades diferentes. Los enlaces de aplicación se organizan en tipos de enlace de aplicación o simplemente tipos de enlace. Los tipos de enlaces se caracterizan por un nombre, un conjunto de fuentes, que indica de dónde puede partirse en la navegación, y un destino, que indica hacia dónde va el enlace. Por último tiene un atributo especial que puede tomar los valores simétrico o asimétrico, para indicar si el enlace es en un único sentido o no.

Como cualquier otro modelo de diseño, HDM distingue entre el concepto de esquema y de instancia del esquema. El esquema define la estructura general de la aplicación y la instancia son las unidades, perspectivas y enlaces concretos que cumplan con los requisitos establecidos en el modelo. En una aplicación multimedia es necesario plantearse, además, lo que HDM define como semántica de navegación. La semántica de navegación representa cómo se va a mostrar la información al usuario. Para ello, HDM define una semántica de navegación por defecto. En ésta se asume que al usuario se le muestra la información mediante nodos o unidades de forma que sólo un nodo está activo en cada momento. Asumiendo esta idea, los usuarios sólo serán conscientes de los enlaces que denominamos de perspectiva. Por ello, es necesario que todas las posibilidades de navegación que se quieran representar en HDM deban estar traducidas a este tipo de enlaces.

En la actualidad HDM no se usa, principalmente por dos razones. Por un lado, el paradigma estructurado ha dejado paso al paradigma Orientado a Objetos y por otro, aunque HDM propone ideas para el desarrollo de aplicaciones multimedia, no ofrece un proceso metodológico completo. Pero, a pesar de que haya caído en desuso, HDM ha servido como base a otras importantes metodologías como son RMM y OOHDM.

Elementos de HDM

El HDM se integra por un conjunto de elementos que cuentan con características de estructura y propósito de acuerdo con la siguiente descripción:

- Ranura, representa una pieza atómica de información con un tipo simple (entero, carácter, etc.) o complejo (video, sonido, etc.); su comportamiento implica el uso de controles para su exhibición y reproducción
- Marco, es un conjunto de ranuras que integran información como una unidad a ser presentada en forma pasiva al exhibirse una sola vez y por completo, o bien activamente mediante la sincronía con eventos

- Nodo, es una unidad navegacional que se asocia a un marco, y que puede ser referenciada directa o indirectamente por el usuario
- Tipo de nodo, es una agrupación de nodos con semejantes características los cuales tienen marcos y ligas similares
- Componente, es una unidad lógica de nodos que se exhibe mediante la navegación por medio de ligas que el usuario activa
- Tipo de componente, son elementos de la misma categoría que tienen nodos, dinámicas y conexiones semejantes
- Entidad, es un objeto del mundo real representado por medio de un grupo de componentes, cuya navegación está organizada por una estructura
- Tipo de entidad, es un conjunto de entidades con estructuras y conexiones similares
- Colección es un conjunto de objetos que representan entidades, componentes, nodos y marcos; con el propósito de caracterizar una taxioma o facilitar la percepción que se tiene de cierto contenido; su navegación está determinada por medio de una colección de ligas
- Liga, es una conexión entre dos diferentes elementos que constituye la unidad objeto de navegación que de acuerdo con sus características y reglas dan vida a un estilo
- Ligas estructurales, se asocian a nodos de componentes y entidades, reflejando las relaciones lógicas entre los elementos para lo cual emplean una estructura de navegación
- Colección de ligas, puede ser una índice de ligas o una visita guiada (guided tour) la cual asocia la colección de nodos de acuerdo con una estructura lógica, para ello se puede emplear facilidades para atravesar la colección de ligas
- Índice de ligas, asocia la colección de nodos a cada miembro y viceversa, su recorrido se lleva a cabo por medio de un índice
- Visita guiada, es una visita guiada de ligas que conectan a una colección de miembros en forma lineal, secuencial, circular y aleatoria, su comportamiento es activado por el sistema o por el usuario,

- Esquema aplicativo, conecta dos objetos de acuerdo con alguna relación de la aplicación
- Esquema de ligas, asocia dos objetos conforme a un tipo de liga definida relacionada con un tema específico
- Tipo de liga, agrupan a todos los esquemas de liga con un significado semejante, asociando objetos de la misma categoría
- Colección de navegación, son un conjunto de objetos llamados miembros, que es navegada directamente al acceder a un miembro por medio de un índice o bien, moviéndose desde un miembro al otro de acuerdo con un orden
- Liga de navegación estructural y aplicativa, concierne a la exploración de nodos que pertenecen a la misma entidad

2.4. RMM (Relationship Management Methogology)

RMM asume las extensiones que HDM incluye en los clásicos Entidad-Relación y añade un nuevo concepto que denomina slice. Un slice es un subconjunto de atributos de una entidad que van a ser presentados de forma agrupada al usuario. De esta forma, una aplicación estará formada por entidades cuyos atributos son agrupados en slices. En la figura 2 se representan los slices.

Por otro lado, es necesario representar los enlaces. Aunque RMM toma a HDM como base, define sus propios enlaces y los divide en varios grupos:

- Enlaces no condicionales, pueden ser unidireccionales o bidireccionales. Serían los enlaces en los que al partir del origen se pasa al destino sin necesidad de que el usuario indique ninguna condición
- Enlaces condicionales, en ellos el usuario tiene que indicar alguna condición específica. Dependiendo de dicha condición el usuario irá a un slice u otro
- Rutas guiadas, se activan automáticamente ante un evento o pasado un tiempo, el usuario no debe hacer nada

Existe la posibilidad de mezclar varios de estos enlaces. Por ejemplo, se puede optar por varias rutas guiadas dependiendo de las condiciones que introduzca el usuario. En la figura 2 se muestra cómo representarían en el modelo Entidad-Relación estos conceptos.

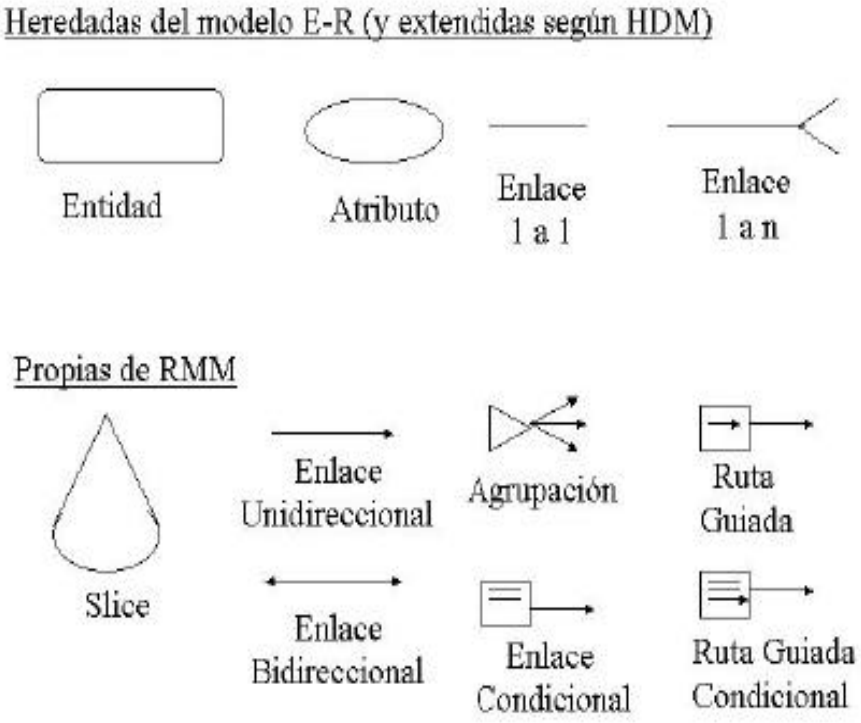


Figura 2. Primitivas del modelo RMM

Basándose en todas estas ideas, en RMM se representa la aplicación mediante un modelo, denominado RMDM. Este modelo es un enriquecimiento del modelo Entidad-Relación y permitirá representar a las aplicaciones multimedia. En este modelo podemos encontrar elementos propios de la propuesta del modelo Entidad-Relación (entidades, atributos, etc.) aunque con las extensiones de HDM y los nuevos conceptos definidos anteriormente (enlaces, rutas guiadas, slice, etc.). En la figura 2 se pueden ver todos los iconos que se pueden encontrar en un modelo RMDM.

En la figura 3 se muestra el ciclo de vida propuesto por RMM.

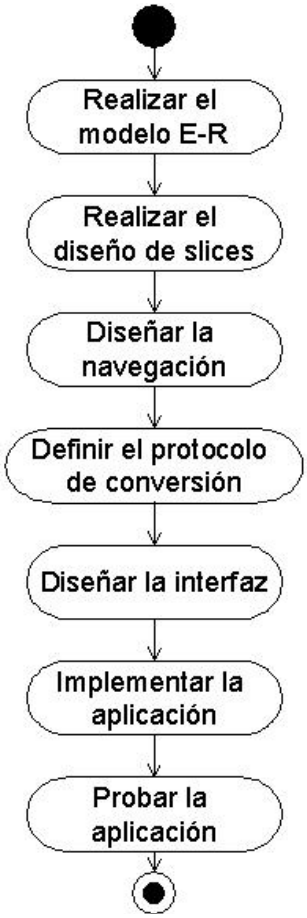


Figura 3. Proceso de desarrollo de RMM

2.4.1. Fases de RMM

Como ya se ha comentado, RMM propone un proceso dividido en etapas para el desarrollo de las aplicaciones multimedia. A continuación se verán muy brevemente estas fases y sus objetivos.

Fase 1- Realizar el modelo E-R

En esta fase se debe obtener un modelo Entidad-Relación del sistema, sin necesidad de entrar en detalles de navegación o de presentación al usuario. Se debe actuar de la misma forma que se actuaría para obtener un modelo E-R de una aplicación software clásica.

Fase 2- Realizar los diseños de slice

Para cada entidad detectada en la fase anterior, se debe definir un diagrama de slices. Esto es, se deben detectar los slices para esa entidad, es decir, cómo se van a presentar los atributos de la entidad al usuario. Se debe obtener un modelo compuesto por slices y enlaces (ya sean guiados, direccionales, etc.), mediante la representación que se mostró en la figura 2.

Fase 3- Diseñar la navegación

Una vez que ya se han definido los slices, se debe diseñar cómo se pasará de una entidad a otra, es decir, hay que enriquecer el modelo Entidad-Relación obtenido en la primera fase con los enlaces entre entidades. El modelo RMDM de la aplicación será la unión del resultado de la fase 3 y de la fase 2.

Fase 4- Definir el protocolo de conversión

En esta fase se debe definir el proceso a seguir para pasar del modelo RMDM a la plataforma de desarrollo concreta. En principio no se propone ninguna técnica estándar a seguir para ello.

Fase 5- Diseñar la interfaz

En esta fase se diseñan las pantallas tal y como se van a mostrar al usuario. Por regla general cada slice se va a corresponder con una pantalla. En esta fase ya es necesario entrar en aspectos concretos del lenguaje de programación que se va a usar.

Fase 6- Implementar la aplicación

En base al protocolo establecido en la fase 4 y al modelo RMDM obtenido, se implementa el sistema.

Fase 7- Probar la aplicación

Una vez que se obtiene la aplicación ejecutable, se deben realizar las pruebas de funcionamiento a la misma. Para ello es necesario definir el test de prueba y estudiar sus resultados.

2.5. EORM (Enhanced Object Relationship Methodology)

EORM es una metodología sencilla, que siendo orientada a objetos garantiza todas las ventajas que el orientado a objetos ofrece, pero además aumenta las posibilidades de reutilización en las aplicaciones, gracias a la definición del repositorio o librerías de clases enlace.

La aplicación de la metodología EORM puede resultar bastante sencilla gracias al uso de ODMTool, una herramienta desarrollada por el propio autor de EORM. Esta herramienta se usa de forma conjunta con el generador comercial de interfaces gráficas de usuario, ONTOS Studio y con un sistema gestor de base de datos orientado a objetos, de manera que se permite el diseño interactivo de esquemas EORM y la generación automática de código, inicialmente en C++.

EORM es la primera propuesta orientada a objetos.

En la figura 4 se muestra el ciclo de vida propuesto por EORM.

Figura 4. Proceso de desarrollo de EORM

2.5.1. Fases de EORM

Fase 1- Análisis

Realmente esta fase no puede denominarse análisis. Se correspondería más a un diseño de objetos, hablando en términos de OMT (RUMBAUGH 1991). Consiste en hacer un modelo orientado a objetos, según las pautas y nomenclatura de OMT para representar la aplicación. OMT es un lenguaje de modelado que ha servido como base para UML (GRADY BOOCH 1999) y el Proceso Unificado (IVAR JACOBSON 1999).

En esta primera etapa de análisis de EORM no se tendrán en cuenta aspectos como la navegación o la interfaz, dejándose ambas para etapas posteriores.

Fase 2- Diseño

Durante el diseño se procede a modificar el modelo de objetos obtenido en la fase anterior añadiendo semántica suficiente a las relaciones para representar los enlaces. Este modelo de objetos enriquecido se denomina EORM y en él se van a reflejar tanto la estructura de la

información (modelo abstracto hipermedial compuesto por nodos y enlaces) como las posibilidades de navegación ofrecidas por el sistema. Para recoger esto último, existirá un repositorio o librería de clases de enlaces, donde se especificarán las posibles operaciones asociadas a cada enlace de un hiperdocumento. Estas operaciones serán del tipo crear, eliminar, siguiente, etc. Pero además en estas librerías se recogerán los atributos del enlace: fecha de creación, forma de presentación en la pantalla, etc.

Para estas librerías los autores de EORM dan una serie de tipos de enlaces ya predefinidos. A continuación se describen de forma sencilla.

- simpleLink: es un enlace dentro del mismo documento o la misma página
- navigationalLink: es un mecanismo que representa un hiperenlace. Esta clase posee una función de backtracking que permite volver a la página origen
- nodeToNode: es un enlace que une dos clases del modelo
- spanToNode: este enlace une el contenido de un objeto con otro objeto
- structureLink: es un conjunto de enlaces simples, por ejemplo, representa un menú
- setLink: es un structureLink que da acceso a una colección de objetos sin importar el orden
- listLink: es un structureLink que da acceso a una colección de objetos según un determinado orden

Fase 3- Construcción

En esta fase se prepararía el código fuente para cada una de las clases y la interfaz gráfica de usuarios. No se da ninguna recomendación especial para ello.

2.6. OOHDM (Object-Oriented Hypermedia Design Method)

Esta metodología presenta varias características y es necesario resaltar algunas de ellas. La primera de ellas como ya se había planteado anteriormente es orientada a objetos. En esto

se diferencia de su antecesor HDM. A la hora de la representación de las clases y los diagramas, OOHDM utiliza la nomenclatura propuesta por OMT (RUMBAUGH 1991). Sin embargo en las últimas versiones ya se ha asumido UML.

Otra característica de OOHDM es que, a diferencia de HDM, no sólo propone un modelo para representar a las aplicaciones multimedia, sino que propone un proceso predeterminado para el que indica las actividades a realizar y los productos que se deben obtener en cada fase del desarrollo.

Fundamentalmente OOHDM toma como partida el modelo de clases que se obtiene en la fase de diseño de objetos de OMT o, en sus últimas versiones, en el análisis del Proceso Unificado de UML. A este modelo lo denomina modelo conceptual. El proceso anterior a éste, es decir, el análisis y el diseño de sistemas, sería idéntico al que se realizaría en el desarrollo de un sistema clásico según OMT o UML.

Partiendo de este modelo conceptual, OOHDM propone ir añadiendo características que permitan incorporar a esta representación del sistema todos los aspectos propios de las aplicaciones multimedia. En una segunda etapa de diseño, se parte de ese modelo conceptual y se añade a éste todos los aspectos de navegación, obteniéndose un nuevo modelo de clases denominado modelo navegacional. Por último, este modelo sirve como base para definir lo que en el argot de OOHDM se denomina modelo de interfaz abstracta. El modelo de interfaz abstracta representa la visión que del sistema tendrá cada usuario del mismo.

Lo más interesante de OOHDM es la gran aceptación que ha tenido el proceso de desarrollo que propone y que se muestra en la figura 5.

Figura 5. Proceso de desarrollo de OOHDM

2.6.1. Fases de OOHDM

OOHDM es una mezcla de estilos de desarrollo basado en prototipos, en desarrollo interactivo y de desarrollo incremental. En cada fase se elabora un modelo orientado a objetos conceptual que recoge las características a resaltar en la misma incrementando los resultados de la fase o fases anteriores.

Fase 1- Diseño Conceptual

Se construye un modelo orientado a objetos que represente el dominio de la aplicación usando las técnicas propias de la orientación a objetos. La finalidad principal durante esta fase es capturar el dominio semántico de la aplicación en la medida de lo posible, teniendo en cuenta el papel de los usuarios y las tareas que desarrollan. El resultado de esta fase es un modelo de clases relacionadas que se divide en subsistemas. En la tabla 1 se esquematiza esta fase.

Fase	**Diseño conceptual**
Productos	Diagrama de Clases, División en subsistemas y relaciones
Herramienta	Técnicas de modelado O.O, patrones de diseño
Mecanismos	Clasificación, agregación, generalización y especialización
Objetivo de diseño	Modelo semántico de la aplicación

TABLA 1: FASE DE DISEÑO CONCEPTUAL DE OOHDM

Fase 2- Diseño Navegacional

En OOHDM una aplicación se ve a través de un sistema de navegación. En la fase de diseño navegacional se debe diseñar la aplicación teniendo en cuenta las tareas que el usuario va a realizar sobre el sistema. Para ello, hay que partir del esquema conceptual desarrollado en la fase anterior. Hay que tener en cuenta que sobre un mismo esquema conceptual se pueden desarrollar diferentes modelos navegacionales (cada uno de los cuales dará origen a una aplicación diferente).

La estructura de navegación de una aplicación multimedia está definida por un esquema de clases de navegación específica, que refleja una posible vista elegida. En OOHDM hay una serie de clases especiales predefinidas, que se conocen como clases navegacionales: *Nodos, Enlaces y Estructuras de acceso*, que se organizan dentro de un *Contexto Navegacional*. Mientras que la semántica de los nodos y los enlaces son comunes a todas las aplicaciones hipermedia, las estructuras de acceso representan diferentes modos de acceso a esos nodos y enlaces de forma específica en cada aplicación.

1- Nodos: Los nodos son contenedores básicos de información de las aplicaciones hipermedia. Se definen como vistas orientadas a objeto de las clases definidas durante el diseño conceptual usando un lenguaje predefinido y muy intuitivo, permitiendo así que un nodo sea definido mediante la combinación de atributos de clases diferentes relacionadas en el modelo de diseño conceptual. Los nodos contendrán atributos de tipos básicos (donde se pueden encontrar tipos como imágenes o sonidos) y enlaces.

2- Enlaces: Los enlaces reflejan la relación de navegación que puede explorar el usuario. Se sabe que para un mismo esquema conceptual puede haber diferentes esquemas navegacionales y los enlaces van a ser imprescindibles para poder crear esas vistas diferentes.

OOHDM hace una definición de clase enlace que contiene un único atributo. Lo vemos en la figura 6.

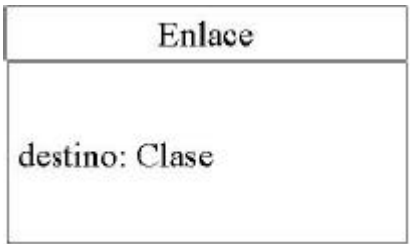


Figura 6. Definición de la clase enlace

Su atributo almacenaría la clase a la que se navega por ese enlace. Las clases enlaces sirven para especificar los atributos de enlaces y estos a su vez para representar enlaces entre clases nodos o incluso entre otros enlaces. En cualquier caso, el enlace puede actuar como un objeto intermedio en un proceso de navegación o como un puente de conexión entre dos nodos.

3- Estructuras de Acceso: Las estructuras de acceso actúan como índices o diccionarios que permiten al usuario encontrar de forma rápida y eficiente la información deseada. Los menús, los índices o las guías de ruta son ejemplos de estas estructuras. Las estructuras de acceso

también se modelan como clases, compuestas por un conjunto de referencias a objetos que son accesibles desde ella y una serie de criterios de clasificación de las mismas.

4- Contexto Navegacional: Para diseñar bien una aplicación hipermedia, hay que prever los caminos que el usuario puede seguir, así es como únicamente se podría evitar información redundante o que el usuario se pierda en la navegación. En OOHDM un contexto navegacional está compuesto por un conjunto de nodos, de enlaces, de clases de contexto y de otros contextos navegacionales. Estos son introducidos desde clases de navegación (enlaces, nodos o estructuras de acceso), pudiendo ser definidas por extensión o de forma implícita.

5- Clase de Contexto: Es otra clase especial que sirve para complementar la definición de una clase de navegación. Por ejemplo, sirve para indicar qué información está accesible desde un enlace y desde dónde se puede llegar a él.

TRANSFORMACIÓN NAVEGACIONAL:

Cuando la aplicación se ejecuta en una sola vista, el contexto navegacional tiene bastante poder semántico como para representar la aplicación. Sin embargo, cuando pueden aparecer diferentes contextos de navegación a la vez, se requiere el uso de transformaciones navegacionales. A través de ellos podemos expresar concurrencias o navegaciones que se ejecutan a la par. En la tabla 2 vemos un resumen de esta fase.

Fase	**Diseño navegacional**
Productos	Nodos, enlaces, estructuras de accesos, contextos navegacionales y transformaciones navegacionales
Herramienta	Técnicas de modelado O.O, patrones de diseño, diagramas de estados, escenarios
Mecanismos	Clasificación, agregación, generalización y especialización
Objetivo de diseño	Establecer los recorridos que el usuario puede seguir por la aplicación

TABLA 2: FASE DE DISEÑO NAVEGACIONAL DE OOHDM

Fase 3- Diseño de Interfaz Abstracta

Una vez definida la estructura navegacional, hay que prepararla para que sea perceptible por el usuario y esto es lo que se intenta en esta fase. Esto consiste en definir qué objetos de interfaz va a percibir el usuario, y en particular el camino en el cuál aparecerán los diferentes objetos de navegación, qué objeto de interfaz actuará en la navegación, la forma de sincronización de los objetos multimedia y el interfaz de transformaciones. Al haber una clara separación entre la fase anterior y esta fase, para un mismo modelo de navegación se pueden definir diferentes modelos de interfaces, permitiendo, así que el interfaz se ajuste mejor a las necesidades del usuario.

MODELOS DE VISTAS ABSTRACTAS DE DATOS (ADVs): los modelos de los ADVs no son más que representaciones formales que se usan para mostrar:

a) La forma en que se estructura la interfaz, para ello se usan las vistas abstractas de datos. Estos son elementos que tienen una forma y un dinamismo. Son elementos abstractos en el sentido de que solo representan la interfaz y su dinamismo, y no la implementación, no entran

en aspectos concretos como el color de la pantalla o la ubicación en ésta de la información. Así, tendremos un conjunto de representaciones gráficas, que gestionan las estructuras de datos y de control, y un conjunto de aspectos de interfaz, como las entradas del usuario y las salidas que se le ofrecen.

b) La forma en que la interfaz se relaciona con las clases navegacionales, para ello se usan diagramas de configuración. Los diagramas de configuración van a ser grafos dirigidos que permitirán indicar de qué objetos de navegación toman la información los ADV.

c) La forma en que la aplicación reacciona a eventos externos, para ello se usan los ADVs-Charts. Los ADVs-Charts van a ser diagramas bastante similares a las máquinas de estados, es más en las últimas versiones de OOHDM se usan máquinas de esto. A través de ellas se puede indicar los eventos que afectan a una ADV y cómo ésta reacciona a ese elemento.

Como resumen, en la tabla 3 vemos descrita esta fase.

Fase	**Diseño de interfaz abstracta**
Productos	Objetos de interfaz abstracta, respuestas a eventos externos y transformaciones de interfaz
Herramienta	ADVs, Diagramas de configuración, ADV-Charts y patrones de diseño
Mecanismos	Mapeado entre los objetos de navegación y los objetos visibles
Objetivo de diseño	Modelado de los objetos perceptibles por el usuario y de cómo le afecta a la aplicación los eventos externos

TABLA 3: FASE DE DISEÑO DE INTERFAZ ABSTRACTO DE OOHDM

Fase 4- Implementación

Una vez obtenido el modelo conceptual, el modelo de navegación y el modelo de interfaz abstracta, sólo queda llevar los objetos a un lenguaje concreto de programación, para obtener

así la implementación ejecutable de la aplicación. En la tabla 4 vemos un resumen de esta fase.

Fase	**Implementación**
Productos	Aplicación ejecutable
Herramienta	El entorno del lenguaje de programación
Mecanismos	Los ofrecidos por el lenguaje
Objetivo de diseño	Obtener la aplicación ejecutable

TABLA 4: FASE DE IMPLEMENTACIÓN DE OOHDM

Para terminar, se podría decir que los puntos claves de OOHDM se encuentran en:

- OOHDM contempla los objetos que representan la navegación como vistas de los objetos detallados en el modelo conceptual
- OOHDM abstrae los conceptos básicos de la navegación: nodos, enlaces e índices y los organiza mediante el uso de los contextos de navegación, permitiendo así una organización adecuada de los mismos
- OOHDM separa las características de interfaz de las características de navegación, con las ventajas que esto supone

2.7. SOHDM (Scenario-based Object-oriented Hypermedia Design Methodology)

Como ya se ha comentado, SOHDM es una metodología para el desarrollo de aplicaciones multimedia que se divide en seis fases que hay que realizar de forma secuencial. Sin embargo, el proceso de desarrollo es un proceso cíclico en el sentido de que al realizar una fase se puede regresar a alguna de las anteriores para refinarla y adaptarla mejor.

Como su propio nombre indica, SOHDM está basado en los escenarios para elaborar las aplicaciones multimedia. En su proceso, los escenarios se elaboran en la fase de análisis para capturar los requisitos funcionales del sistema y sirven como base para el resto del proceso. A pesar de que SOHDM se asemeja bastante a OOHDM y EORM, difiere de ellas en el sentido de que mientras que estas dos metodologías sólo trabajan en la fase de diseño, SOHDM engloba también la fase de análisis.

En la figura 7 se muestra el ciclo de vida propuesto por SOHDM.

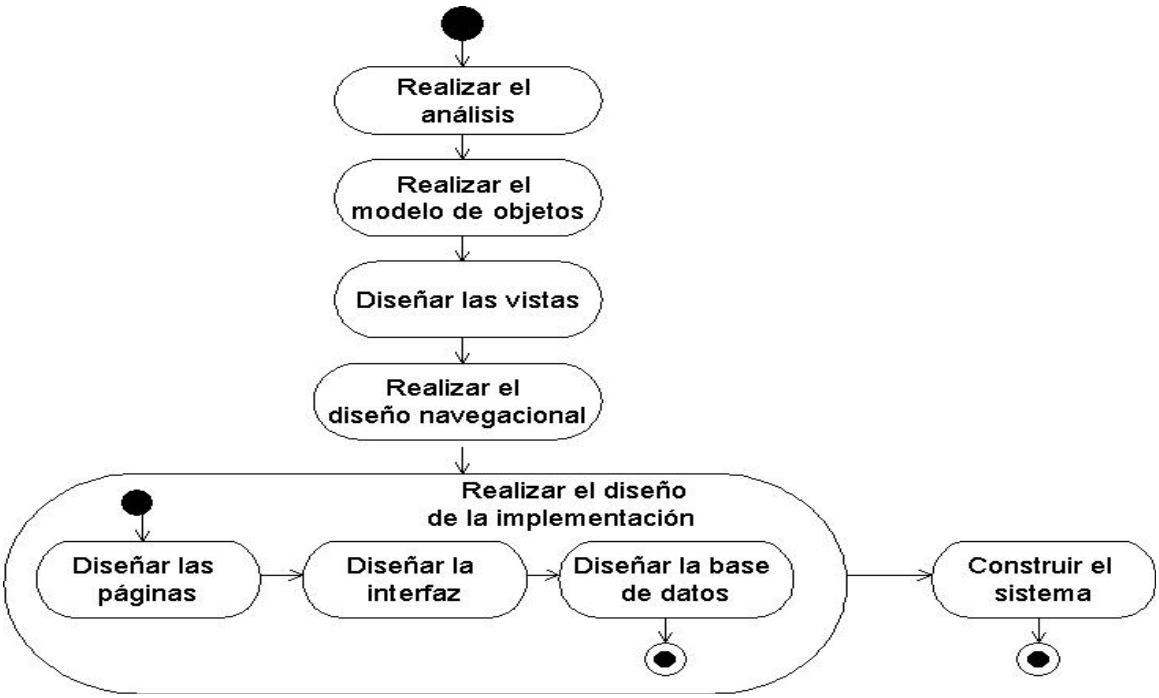


Figura 7: Proceso de desarrollo de SOHDM

2.7.1. Fases de SOHDM

Ya se ha comentado que SOHDM es un proceso compuesto por seis fases secuenciales.

Fase 1- Análisis

En la fase de análisis se debe realizar un estudio de las necesidades de la aplicación, del entorno de trabajo y de los actores. La finalidad principal de esta fase es conseguir los escenarios que representen las actividades que se pueden llevar a cabo en el sistema.

Para ello, lo primero que se debe realizar es un diagrama de contexto tal y como se propone en diagramas de flujos de datos (YOURDON 1989). En este diagrama de contexto es necesario detectar a las entidades externas que se comunican con el sistema, así como los eventos que provocan esa comunicación.

Una vez detectados estos eventos, se debe elaborar lo que se denomina lista de eventos. La lista de eventos será una tabla con una estructura similar a la mostrada en la tabla 5.

Nombre de la entidad externa	**Nombre del evento**
Entidad externa 1	Evento 1 en el que participa la entidad
	…
	Evento n en el que participa la entidad
…	…
Entidad externa m	Evento 1 en el que participa la entidad
	…
	Evento p en el que participa la entidad

TABLA 5: ESTRUCTURA DE LA LISTA DE EVENTOS

En esta tabla se detallan todas las entidades en la columna de la izquierda y en la columna de la derecha todos los eventos en los que cada una participa.

Partiendo de esta tabla, por cada evento diferente se debe elaborar un escenario. Éstos se van a representar mediante los denominados SACs (Scenario Activity Chart). En estos escenarios se va a describir el proceso de trabajo que se va a seguir en el sistema cuando se

produzca la situación que el escenario representa. Por ejemplo, en la figura 8 se encuentra un escenario en el que se describe cómo se produce la conexión al sistema del usuario.

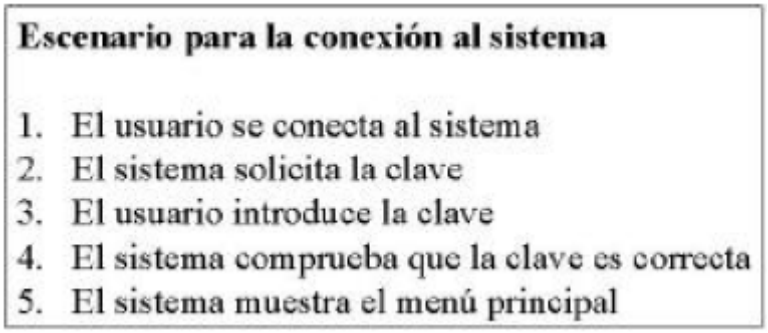


Figura 8. Ejemplo de escenario

Fase 2- Modelado de objetos

Los escenarios representados mediante SACs son usados para conseguir modelar los objetos del sistema. En la fase de modelado de objetos, los escenarios van a ser transformados en objetos según la propuesta de los CRC Cards (Class Responsability Collaboration) (REBECCA WIRFS-BROCK 1990). Esta propuesta tiene como objetivo presentar un formato sencillo e informal para conseguir un diccionario de datos para las clases del sistema. En ellas cada clase tiene asociado una ficha (CRC Cards) en la que se almacena: su nombre, sus atributos, su superclase, sus subclases, sus componentes, las asociaciones en las que participa, las otras clases con las que colabora y los eventos detallados en los SACs de los que es responsable.

Las CRC Cards irán acompañadas de un diagrama de clases, CSD (Class Structure Diagram), que representará gráficamente lo recogido en las CRC Cards. En principio la nomenclatura seguida fue la de OMT pero ya ha asumido la de UML.

Fase 3- Diseño de vistas

En la fase de diseño de vistas, los objetos serán reorganizados en unidades navegacionales. Una unidad navegacional representa una vista de los objetos del sistema. Las vistas van a estar muy relacionadas con el concepto de nodos de OOHDM.

La vista es una agrupación de información que se presenta agrupada al usuario bajo un determinado criterio. En SOHDM, las vistas pueden ser:

- Vistas base; son aquellas que toman todos los datos que muestra de una única clase
- Vistas de asociación; toma los datos de dos clases que se encuentran relacionadas mediante una asociación en el modelo de clases
- Vistas de colaboración; toma los datos de clases que se encuentran relacionadas mediante una relación de colaboración

Fase 4- Diseño Navegacional

En el diseño navegacional se van a definir los enlaces o hiperenlaces que existen entre las diferentes vistas. Aquí las vistas definidas en la fase anterior se relacionan a través de estructuras de acceso.

El primer paso a realizar es definir lo que se conoce como nodos de estructuras de acceso (ASNs). Estos son nodos especiales como diccionarios, menús, etc. que sirven como punto de partida para la navegación. Una vez definidos los ASN, estos y las vistas definidas en la fase anterior, se conectan mediante flechas que indican el sentido de la navegación. Se obtiene así un grafo en el que se representa como se navega desde un punto de información (vista o ASN) a otro. A este modelo se le denomina enlaces navegacionales.

A pesar de que el grafo que surge es bastante intuitivo, SOHDM propone una alternativa para representar los enlaces navegacionales. Por cada componente conexa de este grafo, se elabora una matriz, denominada matriz de enlaces navegacionales. En esta matriz se indica como los nodos de ese componente se relacionan entre sí.

Fase 5- Diseño de la implementación

En esta fase se van a generar esquemas de páginas que van a representar los puntos de información definidos en la fase anterior dentro de un entorno determinado. Para cada esquema se debe indicar: su nombre, su título, las vistas que engloba, una breve descripción de su significado y una lista con los enlaces que tiene.

Tras definir estos puntos de información se hace un diseño de la interfaz de usuario. SOHDM tiene prevista una nomenclatura normalizada para representar los posibles elementos que se pueden encontrar en una pantalla: botones, imágenes, listas, etc.

Una vez definida la interfaz de usuario y los esquemas, es necesario definir la base de datos. En principio las aplicaciones hipermedia deben definirse en sistemas gestores orientados a objeto, pero, acercándose más a la realidad, permite el uso de sistemas gestores de bases de datos relacionales. Para poder llevar una representación orientada a objetos a un modelo relacional, es necesario aplicar técnicas de conversión.

Fase 6- Construcción

En la fase de construcción se debe implementar una aplicación hipermedia ejecutable en función de las pantallas y las páginas definidas en la fase anterior. Igualmente debe desarrollarse la base de datos física para soportar la aplicación.

2.8. HFPM (Hypermedia Flexible Process Modeling Strategy)

A continuación, en la figura 9 se muestra el ciclo de vida propuesto por HFPM.

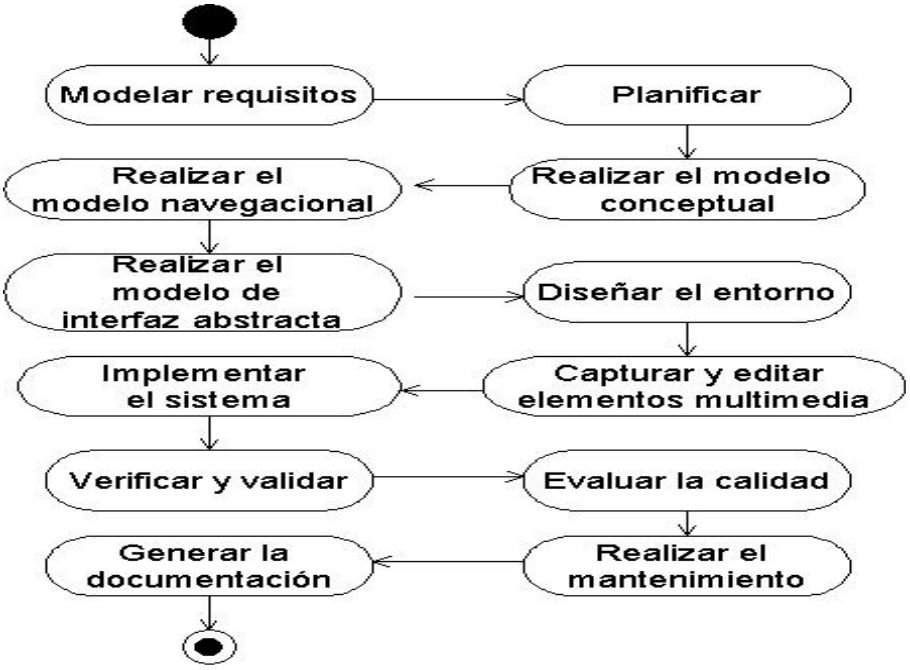


Figura 9. Proceso de desarrollo de HFPM

2.8.1. Fases de HFPM

Fase 1- Modelado de los requisitos del software

En esta fase se proponen las siguientes tareas:

- Descripción breve del problema
- Descripción de los requisitos funcionales mediante los casos de uso
- Realizar un modelo de datos para esos casos de uso
- Modelar la interfaz de usuario
- Modelar los requisitos no funcionales. En éstos incluyen la navegación, la seguridad, etc.

En principio no da ninguna norma a seguir para realizar estas tareas, dejando flexible las representaciones.

Fase 2- Planificación

En esta fase se plantea el plan de trabajo del proyecto, se analiza y se especifica de forma concisa. No se presenta ningún patrón o proceso a seguir para ello.

Fase 3- Modelado conceptual

El objetivo de esta fase es similar al objetivo de la fase de modelado conceptual de OOHDM. En esta fase se debe conseguir un modelo de clases que represente al sistema sin entrar en aspectos de hipermedia. Las tareas a realizar son:

- Analizar el dominio del problema
- Modelar el sistema mediante un diagrama de clases

Fase 4- Modelado Navegacional

Está íntimamente basado en las propuestas de OOHDM y EORM. En esta fase se trata de conseguir un modelo navegacional que represente las posibilidades de navegación del sistema. Para ello, es necesario:

- Analizar las necesidades de los usuarios
- Identificar las clases navegacionales, entendiendo por clases navegacionales las propuestas por OOHDM
- Especificar el esquema de navegación y las clases de navegación
- Analizar los contextos de navegación
- Especificar los contextos navegacionales

Fase 5- Modelado de la Interfaz Abstracta

Consiste en hacer un diseño de la interfaz sin entrar en características propias del lenguaje de programación. Las tareas que se proponen son:

- Analizar las necesidades de interfaz de usuario

- Detectar los objetos, los eventos y los enlaces visibles para cada usuario
- Diseñar los prototipos

Fase 6- Diseño del Entorno

Esta fase tiene por objetivos enriquecer los modelos obtenidos mediante el uso de patrones de diseño. Además es en esta fase en la que se decide la arquitectura del sistema y la división en subsistemas. Las tareas son las siguientes:

- Usar patrones de diseño para mejorar los modelos
- Diseñar la arquitectura, si es posible haciendo uso de patrones
- Dividir el sistema en subsistemas

Fase 7- Capturar y editar los elementos multimedia

En esta fase se deben plantear los múltiples medios con los que se va a trabajar, así como los sistemas de almacenamiento que se usarán en los mismos.

Fase 8- Implementación

Esta fase tiene por objetivo conseguir el programa ejecutable que represente la aplicación. Al igual que la fase anterior no tiene tareas.

Fase 9- Verificación y validación

En esta fase se va a analizar si el resultado es adecuado en base a los requisitos estudiados en la primera fase del proceso. No se dan pautas a seguir para ello dentro de la propuesta de HFPM.

Fase 10- Evaluación del entorno

Aquí se analiza si el resultado se adecua al entorno en el que se va a implantar.

Fase 11- Evaluación de la calidad

Se debe evaluar y determinar si el producto resultante es de alta calidad. En este aspecto HFPM no especifica qué entiende por calidad ni cómo se puede evaluar la calidad del producto.

Fase 12- Mantenimiento

En general esto implica tanto el mantenimiento correctivo, aumentativo y adaptativo del sistema, aunque nuevamente no se dan pautas a seguir para ello.

Fase 13- Documentación

Como resultado final se debe generar la documentación del sistema. Dicha documentación engloba:

- La documentación de cada uno de los modelos generados en cada fase
- La documentación del resultado de las pruebas y las valoraciones realizadas en las fases 9, 10 y 11
- El manual de usuario

Aunque la presentación de las fases es secuencial, HFPM permite la vuelta atrás y la realización de iteraciones en el proceso de desarrollo.

2.9. OO/Pattern Approach

Esta metodología es bastante completa en el sentido de que hace referencia a la fase de análisis, diseño e implementación. Resalta la necesidad de realizar un diccionario de datos para el modelo conceptual.

Algo característico de esta propuesta es el hecho de que utiliza los conocidos casos de uso para realizar la fase de análisis de la aplicación.

2.9.1. Fases de OO/Pattern Approach

Fase 1- Diseño de los casos de uso

Basándose en la técnica de los casos de uso, tal y como la propone UML (IVAR JACOBSON 1999), esta metodología propone capturar los requerimientos del sistema que servirán para realizar el posterior diseño de la aplicación.

Fase 2- Diseño conceptual

Partiendo de los casos de uso, se debe modelar la aplicación sin entrar en aspectos de interfaz o de navegación. Se pretende conseguir un modelo de clases que represente al sistema. Para ello, se deben aplicar las técnicas orientada a objetos.

Fase 3- Diseño de colaboración

En esta fase es necesario aplicar la técnica de los diagramas de colaboración propuesta por UML (IVAR JACOBSON 1999), para representar las relaciones entre las clases del sistema.

Fase 4- Realizar el diccionario de datos para modelo de clases

En esta fase se debe documentar los modelos obtenidos en las dos fases anteriores. En principio el formato de este diccionario es libre.

Fase 5- Diseño Navegacional

En esta fase se debe enriquecer el modelo de clases con nuevas clases que permitan representar la navegación. Para ello, se definen una serie de patrones de diseño en los que se definen los nodos y los enlaces de forma similar a como se definieron en EORM o OOHDM. El resultado sería un modelo de clases navegacionales similar al de OOHDM.

Fase 6- Implementación

La fase de implementación tiene por objetivo el conseguir, basándose en el modelo navegacional, la aplicación ejecutable del sistema.

2.10. RUP (Rational Unified Process)

RUP divide en 4 fases el desarrollo del software. Cada Fase tiene definido un conjunto de objetivos y un punto de control especifico. A saber:

Fase	**Objetivos**	**Puntos de Control**
Inicio	• Definir el alcance del proyecto • Entender que se va a construir	Objetivo del proyecto
Elaboración	• Construir una versión ejecutable de la arquitectura de la aplicación • Entender cómo se va a construir	Arquitectura de la Aplicación
Construcción	• Completar el esqueleto de la Aplicación con la funcionalidad • Construir una versión Beta	Versión Operativa Inicial de la Aplicación

Transición	• Dar disponibilidad a la aplicación para los usuarios finales • Construir la versión Final	Liberación de la versión de la Aplicación

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

Disciplina de Desarrollo

- *Modelamiento del negocio:* Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- *Requerimientos:* Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- *Análisis y diseño:* Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por lo que indica con precisión lo que se debe programar.
- *Implementación:* Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- *Prueba:* Busca los defectos a los largo del ciclo de vida.

- *Instalación:* Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.

Disciplina de Soporte

- *Gestión de Configuración del Software:* Describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, etc.
- *Gestión de proyecto:* Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.
- *Ambiente:* Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

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

Los elementos de RUP son:

- *Actividades (cómo).* Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- *Trabajadores (quién).* Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- *Artefactos (qué).* Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.

- *Flujos de trabajo (cuándo).* Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

2.10.1. Descripción de las fases y flujos de trabajo de RUP

Fase 1- Inicio

En esta fase se le dedica mayor esfuerzo a los flujos de trabajo de *Modelamiento del negocio y Requerimientos.*

Modelamiento del negocio

Este proceso permite obtener una visión de la organización que permita definir los procesos, roles y responsabilidades de la organización en los modelos de casos de uso del negocio y de objetos. De aquí que este proceso esté relacionado con los de obtención de requerimientos y análisis-diseño.

Los artefactos que se producen en este flujo de trabajo son los siguientes:

- Actor
- Caso de uso
- Modelo de Casos de usos del negocio
- Diagrama de actividades
- Glosario de términos
- Reglas del Negocio
- Listado de riesgos
- Documento Visión

Si se logra determinar el proceso del negocio con fronteras bien establecidas donde se logren ver claramente, quienes son las personas que lo inician, quienes son los beneficiados con cada uno de estos procesos, pero además quienes son las personas que desarrollan las actividades en cada uno de estos procesos, entonces se podrá realizar la *modelación del negocio*. Sino se logra lo planteado en el modelo del negocio entonces identifico conceptos,

se le da definiciones a estos conceptos y se trata de unir o relacionar en otro modelo distinto que es el de *dominio*.

Requerimientos

El modelamiento del negocio brinda una vía natural para determinar los requerimientos del sistema de información.

Un requerimiento es una condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo. Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen. Es una característica que un sistema debe tener para cubrir alguna de las necesidades que lo motivan.

Existen dos grandes tipos de requerimientos:

Requerimientos funcionales

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir.

Requerimientos no funcionales

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

Las actividades que engloba este flujo de trabajo son identificar y clasificar requerimientos, encontrar actores y casos de uso, priorizar casos de uso, detallar casos de uso, prototipar la interfaz de usuario, estructurar el modelo de casos de uso.

Los artefactos que se crean en este flujo de trabajo son:

- Modelo de casos de uso
- Actor
- Caso de uso
- Descripción de la arquitectura (vista del modelo de casos de uso)

- Glosario de términos
- Prototipo de interfaz usuario

Fase 2- Elaboración

En esta fase se le dedica mayor esfuerzo a los flujos de trabajo de *Análisis y Diseño, Implementación* y *Prueba.*

Análisis y Diseño

Este flujo de trabajo tiene como objetivo de trabajo traducir los requisitos a una especificación que describe cómo implementar el sistema.

El análisis consiste en obtener una visión del sistema que se preocupa de ver QUÉ hace, de modo que sólo se interesa por los requisitos funcionales.

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, en definitiva CÓMO cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. De hecho, cuando la precisión del diseño es muy grande, la implementación puede ser hecha por un generador automático de código.

Al final de la fase de inicio hay que definir una arquitectura candidata:

- Crear un esquema inicial de la arquitectura del sistema.
- Identificar clases de análisis y actualizar las realizaciones de los casos de uso con las interacciones de las clases de análisis.

Durante la fase de elaboración se va:

Refinando esta arquitectura hasta llegar a su forma definitiva. En esta fase se desarrolla el proceso por iteraciones, en cada iteración hay que analizar el comportamiento para diseñar componentes. Además si el sistema usará una base de datos, habrá que diseñarla también, obteniendo un modelo de datos. El resultado final más importante de este flujo de trabajo será el modelo de diseño. Consiste en colaboraciones de clases, que pueden ser agregadas en

paquetes y subsistemas. Otro producto importante de este flujo es la documentación de la arquitectura software, que captura varias visiones arquitectónicas del sistema.

RUP define el análisis y el diseño como un único flujo de trabajo en el que hay actividades que se realizan desde la fase de Inicio.

En este flujo de trabajo se generan varios artefactos, en la parte del análisis son:

- Modelo de análisis
- Clases de análisis
- Realización de casos de uso del análisis
- Paquete de análisis
- Descripción de la arquitectura

Los artefactos que se crean en la parte del diseño son:

- Modelo de Diseño
- Modelo de Despliegue
- Clase del diseño
- Interfaz
- Subsistema de diseño
- Realización caso de uso-diseño

Implementación

En este flujo de trabajo empezamos con el resultado del diseño e implementamos el sistema en términos de componentes.

El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue.

El flujo de implementación esta fuertemente determinado por el lenguaje de programación.
Los *diagramas de despliegue* y *componentes* conforman lo que se conoce como un modelo de implementación al describir los componentes a construir y su organización y dependencia entre nodos físicos en los que funcionará la aplicación.
Los artefactos que se generan durante este flujo de trabajo son:

- Modelo de Implementación
- Documento de Arquitectura de Software (actualizado)
- Plan de Integración

Las principales actividades que se realizan en este flujo de trabajo son estructurar el Modelo de Implementación, planificar la Integración e implementar Componentes.

Prueba

Las prueba es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos específicos, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente.
Este flujo de trabajo va a tener como objetivos encontrar y documentar los defectos que puedan afectar la calidad del software, validar que el software trabaje como fue diseñado, validar y probar los requisitos que debe cumplir el software y validar que los requisitos fueron implementados correctamente.
Las actividades que se realizan en este flujo de trabajo son definir la misión de la evaluación, verificar el enfoque de prueba, validar la estabilidad de build, probar y evaluar, lograr la misión aceptable, mejorar la calidad de las pruebas. En las tres primeras actividades se elaboran una serie de artefactos pero los más importantes en este flujo de trabajo son:

- El plan de Pruebas.
- La estrategia de prueba.

- Los casos de pruebas.
- Los Procedimientos de prueba.
- El listado de ideas de prueba.
- El listado de datos de prueba.
- El resumen de la evaluación de las pruebas.

Fase 3- Construcción

En esta fase se le dedica mayor esfuerzo a los flujos de trabajo de *Implementación* y *Prueba.*

Implementación

La implementación es el centro durante las iteraciones de construcción, aunque también se lleva a cabo el trabajo de implementación durante la fase de elaboración, para crear la línea base ejecutable de la arquitectura, y durante la fase de transición, para tratar defectos tardíos como los encontrados con distribuciones beta del sistema.

En esta fase la implementación tendrá los mismos objetivos que tenia en la fase de elaboración. Las actividades siguen siendo las mismas agregándosele dos más, estas son integrar cada subsistema e integrar el sistema.

Los artefactos no siguen siendo los mismos en esta fase los que se generan son los siguientes:

- Componente
- Interfaz
- Subsistema de implementación
- Diagrama de Componentes

Prueba

En esta fase este flujo de trabajo se desarrolla exactamente igual que en la fase de elaboración.

Fase 4- Transición

En esta fase el release ya está listo para su instalación en las condiciones reales. Esto puede implicar reparación de errores. Por tanto el flujo de trabajo que más se desarrolla en esta fase es Instalación realizando una serie de actividades antes mencionadas para logar una eficiente entrega a los clientes del software.

Los tres últimos flujos de trabajo, *Gestión de Configuración del Software, Gestión de proyecto y Ambiente* se enmarcan en todo el ciclo de vida del proyecto desde que este se inicia hasta que se cierra a diferencia de los anteriores que a pesar de que se realizan también en todo el ciclo de vida, tienen un momento esencial en alguna de las fases.

A continuación se expone una breve explicación de dichos flujos de trabajo:

Gestión de Configuración del Software (GCS)

GCS es una disciplina que realiza un control del producto en el proceso de desarrollo del software y su objetivo es el de controlar los cambios. Su propósito es el de controlar los diferentes artefactos que son producidos o elaborados por muchas personas que participan en un mismo proyecto. Mediante esta disciplina se identifica, controla, audita e informa de las modificaciones que invariablemente se dan al desarrollar el software.

Los artefactos más importantes de esta disciplina son:

- Plan de Gestión de la Configuración.
- La solicitud de Cambio.

Gestión de proyecto

RUP incluye dentro de sus disciplinas o flujos de trabajo a la gestión de proyectos, viéndola desde un punto de vista encaminado hacia la planificación y el control de las iteraciones y la gestión de los riesgos en que puede verse involucrado un proyecto.

La aplicación de buenas prácticas en el proceso de desarrollo de software refuerza las posibilidades de éxito que se pueden poseer, por lo que los líderes de los proyectos, junto a sus equipos de trabajo son los responsables de determinar exactamente cómo desarrollar el producto y con qué rigor establecer los principios que se recomiendan.

Los artefactos que se elaboran en este flujo de trabajo son:

- Plan de mitigación de Riesgos
- Plan de aceptación del producto
- Plan de resolución de problemas
- Plan de aseguramiento de la calidad
- Plan de Desarrollo del software

Ambiente

En este flujo de trabajo se realizan una serie de actividades para ir administrando el ambiente de desarrollo.

2.10.2. Extensión de UML para multimedia

Como antes se había planteado una de las principales prácticas que toma en cuanta RUP es que modela el software visualmente, mediante UML. Como una propuesta de extensión de UML se lanza **OMMMA-L** para la integración de especificaciones de sistemas multimedia basados en el paradigma orientado a objetos, y MVC (Modelo Vista Controlador) para la interfaz de usuario.

El MVC es un patrón de diseño de software que distingue un componente modelo sosteniendo la funcionalidad del núcleo y los datos, un componente vista para mostrar la información al usuario y un componente controlador para manipular los eventos de interacción. Separando así los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista pueden ser hechas con un mínimo impacto en el componente del modelo de datos. Un mecanismo de propagación de cambios asegura la consistencia entre el modelo y la interfaz visual. Extendiendo el paradigma MVC para multimedia a las peculiaridades de comportamiento estático y dinámico identificadas anteriormente, obtenemos MVCmm, sobre el que se basa las especificaciones de OMMMA – L.

OMMMA-L no presenta cambios con respecto a UML en el flujo de requisitos y casos de uso. Pero en el modelo de clases de objetos que es con lo que se crea el modelo del dominio si presenta cambios.

En el **Diagrama de clases del modelo de objeto** se agregan 3 conceptos. Un objeto puede ser del tipo:

- *escenario*: cuando representa un conjunto de pantallas que muestran una información a través de objetos con similar funcionalidad.
- *aplicación:* cuando agrupa elementos de media y aúna sus funcionalidades como una entidad.
- *media:* cuando se hace referencia a sonido, texto, imágenes, animaciones, video, botones, etc.

En la parte del **Análisis** OMMMA-L presenta un nuevo diagrama, los Diagramas de Presentación:

El **Diagrama de Presentación** sirve, para describir la parte estática del modelo a través de una descripción intuitiva de la distribución espacial de objetos visuales de la interfaz de usuario. Estos diagrama tienen el propósito de declarar las interfaces de usuario con un

conjunto de estructuras delimitadas en tamaño y área, dividiéndose en objetos de visualización (texto, gráfico, video, animación) e interacción (scrolls, barras de menú, botones, campos de entrada y salida, hipertextos con hipervínculos). Estos diagramas de presentación pueden ser divididos en capas virtuales de presentación donde en cada uno de ellas sólo se haga referencia a una clase específica de componentes (por ejemplo, una vista para los objetos de visualización y otra para los de interacción, u otro tipo de división para la representación de los intereses de los desarrolladores).

EL **Diagrama de Clases del Análisis:** utiliza las mismas notaciones que el Diagrama de Clases de UML, pero incorporando las clases correspondientes a las medias. Este debe reflejar la correspondencia con las medias.

En la parte del **Diseño** se representan:

El **Diagrama de clases del diseño:** OMMMA – L propone en cada diagrama de clases elaborado, adicionar la jerarquía de media de la herramienta y enlazar a través de relaciones las clases del tipo correspondientes. Aquí se incorporan las clases correspondientes a las medias: media continua y media discreta, generalizadas en una clase medias.

Diagrama de Estado: Este diagrama es sintácticamente igual al de diagrama de estado de UML, pero semánticamente no, en el orden de unir los controles interactivos y predefinidos, no interrumpidos de los objetos, las acciones internas de estados simples tienen que llevar nombres de diagrama de secuencia en vez de diagramas de estado empotrados; queriendo esto decir que el comportamiento especificado por el diagrama de secuencia se provoca automáticamente cuando se entra al estado correspondiente donde se hace referencia.

Cada objeto escenario será representado a través de un estado con su mismo nombre, un mensaje de cambio de estado representará una interacción del usuario o de objetos que alteren el comportamiento del sistema. Los estados compuestos son detallados en nuevas máquinas de estados o empotrados en sí mismos y se representan con un color más oscuro. Una acción interna de un estado simple enruta hacia un diagrama de secuencia de

comportamiento temporal. Un estado atómico, siempre que lo amerite, es descrito también por este diagrama.

Diagrama de secuencia extendido a partir del diagrama de secuencia de UML. El Diagrama de secuencia modela una secuencia de una presentación predefinida dentro de una escena, donde todos los objetos dentro de un diagrama se relacionan al mismo eje del tiempo. En este diagrama se hace un refinamiento del eje del tiempo con la introducción de marcas de tiempo a través de diferentes tipos de intervalos; marcas de inicio y fin de ejecución que permite soportar su reusabilidad; marcas de activación y desactivación de demoras en objetos de tipo media, posibilitando la modelación de las tolerancias de la variación de las restricciones de sincronización para los objetos media; activación compuesta de objetos media para la agrupación de objetos concurrentemente activos.

OMMA-L contiene variaciones con respecto a UML solo en las fases de *Análisis y Diseño* manteniéndose normal en los otros flujos. Por tanto no presenta variaciones en el siguiente flujo de trabajo de *Implementación*, ni en el de *Pruebas.*

Conclusiones

En este capítulo se realizó la descripción de las metodologías de desarrollo de software multimedia, detallando cada una de sus fases.

CAPITULO 3. ANÁLISIS COMPARATIVO DE LAS METODOLOGÍAS PARA EL DESARROLLO DE SOFTWARE MULTIMEDIA

Introducción

Basados en las características de las metodologías anteriormente descritas se pasará a compararlas. Se eligieron tres aspectos fundamentales, debe ser Orientada a Objetos, que pueda diferenciar los conceptos de diseño básico o diseño conceptual del diseño navegacional y éste de la interfaz y que cubra todo el ciclo de vida del proyecto dentro. La metodología propuesta debe cumplir con dichos aspectos. Las bases de la propuesta se trazaron por el resultado de las entrevistas hechas a los proyectos multimedia de la facultad 8.

3.1. Análisis comparativo

Uno de los aspectos fundamentales que debe cumplir la metodología escogida es que debe ser Orientada a Objetos ya que resulta un paradigma bastante adecuado para el desarrollo del proyecto este tipo debido a las ventajas presenta. Algunas de las ventajas más importantes son:

- Reutilización. Las clases están diseñadas para que se reutilicen en muchos sistemas. Para maximizar la reutilización, las clases se construyen de manera que se puedan adaptar a los otros sistemas. Un objetivo fundamental de las técnicas orientadas a objetos es lograr la reutilización masiva al construir el software.

- Calidad. Los diseños suelen tener mayor calidad, puesto que se integran a partir de componentes probados, que han sido verificados y pulidos varias veces.

- Un diseño más rápido. Las aplicaciones se crean a partir de componentes ya existentes. Muchos de los componentes están construidos de modo que se pueden adaptar para un diseño particular.
- Mejores herramientas CASE. Las herramientas CASE utilizarán las técnicas gráficas para el diseño de las clases y de la interacción entre ellas, para el uso de los objetos existentes adaptados a nuevas aplicaciones. Las herramientas deben facilitar el modelado en términos de eventos, formas de activación, estados de objetos, etc. Las herramientas OO del CASE deben generar un código tan pronto se definan las clases y permitir al diseñador utilizar y probar los métodos recién creados. Las herramientas se deben diseñar de manera que apoyen el máximo de creatividad y una continua afinación del diseño durante la construcción.

Metodologías	**Orientada a Objeto**
RMM	
EORM	**x**
OOHDM	**x**
SOHDM	**x**
HFPM	**x**
OO/Pattern	**x**
Multimet	
CEDISAC	
OMMMA-L	**x**

Quedan retiradas del próximo análisis comparativo las metodologías RMM, que fue la primera metodología para el diseño de multimedia y tiende a basarse en los ERD; CEDISAC que es basada en la praxis del desarrollo del proceso productivo y no se centra en la especificación

de la estructura al nivel de programación, al igual que Multimet. Además, esta última carece de herramientas de sostén para la descripción del proceso de implementación.

Es importante que la metodología propuesta pueda diferenciar los conceptos de diseño básico o diseño conceptual del diseño navegacional puesto que da una mayor independencia entre los dos aspectos y permite la reutilización y facilitar el mantenimiento del producto informático. Por la misma razón también resulta adecuado separar la navegación de la interfaz.

Metodologías	**Separación Conceptual- Navegación**	**Separación Navegación- Interfaz**
EORM	**x**	
OOHDM	**x**	**x**
SOHDM	**x**	**x**
HFPM	**x**	**x**
OO/Pattern	**x**	
OMMMA-L	**x**	**x**

Las metodologías EORM y OO/Pattern también quedan fuera, debido a que no cumplen con este aspecto establecido en el trabajo.

El último aspecto escogido como objeto de comparación y quizás el más importante es el cubrimiento de todas las fases de desarrollo de un proyecto, o al menos las más significativas. Es por eso que a continuación se tomarán las clásicas fases de todo proceso de desarrollo: especificación, análisis, diseño, codificación y pruebas, como objeto de comparación.

Comenzando por la especificación de requisitos. La captura de requisitos es una fase en la que solo HFPM, SOHDM y OMMMA-L hacen referencia. OOHDM asume que las propuestas como OMT serían adecuadas para este flujo de trabajo. Dentro de las técnicas que se ofrecen para realizar la captura de requisitos se ofrecen dos: los diagramas de casos de uso y

los escenarios. Las primeras dos resultan muy apropiadas para recoger los requisitos funcionales del sistema, pero no permiten especificar qué desea ver el usuario o cómo desea verlo. En el caso de OMMMA-L no se establece un modelo de negocio, sino de dominio.

Se pasa ahora a la fase de análisis, donde se encuentran más propuestas que recogen la necesidad de abordar esta fase de una manera más concreta. De las propuestas que encontramos HFPM, SOHDM, y OMMMA-L que se centran en realizar un modelo de clases a alto nivel para representar el modelo conceptual del sistema.

Algo que falta en las propuestas OOHDM y SOHDM es el plantear como necesario el estudio de la arquitectura del sistema, debido a que las aplicaciones multimedia, son sencillas y no es necesario realizar este estudio, a diferencia de OMMMA-L, que incluye un arquitecto que se encarga de esas tareas dentro del desarrollo del proyecto. Algo que también parece adecuado es el hacer uso de los patrones de diseño.

Con respecto a la implementación, son pocas las propuestas que no la referencia. Sin embargo, OOHDM y SOHDM apenas proponen técnicas de implementación. Lo cual se diferencia de OMMMA-L

La fase de prueba es enunciada en HFPM y OMMMA-L con el inconveniente de que en el primer caso no ofrece técnicas o métodos a la hora de aplicarla, a diferencia del segundo, que sí propone distintos métodos de prueba.

Metodologías	**Especific.**	**Análisis**	**Diseño**	**Codificac.**	**Pruebas**
OOHDM			**x**	**x**	
SOHDM	**x**	**x**	**x**	**x**	
HFPM	**x**	**x**			**x**
OMMMA-L	**x**	**x**	**x**	**x**	**x**

En la comparación anterior se han comparado las metodologías según las fases en las que se desarrollan, destacando cual ha tenido un mayor desarrollo en dicha fase y cual podría mejorar. Por lo que se puede concluir que la extensión OMMMA-L para proyectos multimedia es la única propuesta que cubre todas las fases antes expuestas y de una manera muy completa.

La metodología propuesta además de los aspectos antes mencionados, debe estar orientada al proceso, o sea, que debe indicar qué hacer en cada momento del ciclo de vida. Debe estar orientada al producto, es decir, en cada fase se indicará qué hay que obtener.

De manera general, algo ventajoso y que hace interesante a RMM es que propone un proceso estructurado y definido a seguir para el desarrollo de las aplicaciones multimedia. En este proceso, sin embargo, faltan las primeras etapas a tener en cuenta en cualquier proceso de desarrollo software, como la captura de requisitos. Además, el proceso que ofrece es demasiado abierto en sus fases como para considerarse como una herramienta de desarrollo adecuada, puesto que en la única fase en la que indica una técnica es en la que se hace uso del modelo RMDM (fase 2). Las otras fases quedan abiertas a la opción del diseñador.

A EORM se le pueden hacer ciertas críticas. Por un lado, el proceso metodológico que propone resulta insuficiente en muchos casos principalmente porque solo trata de manera específica los aspectos de almacenamiento y navegación, dejando a un lado temas como la funcionalidad del sistema o los aspectos de interfaz. Además, en ningún momento comenta las técnicas a seguir para obtener los modelos que propone o los productos que se deben generar en el desarrollo. EORM también deja a un lado un aspecto muy importante en la mayoría de las aplicaciones: la captura de requisitos. No sólo no ofrece ninguna propuesta sino que no indica ninguna que se pueda usar.

En cuanto a OOHDM, no puede considerarse una metodología en el amplio sentido, ya que, aunque se detalla el proceso a seguir en lo que sería el diseño de la aplicación, no toma parte en otras fases como pueden ser la captura de requisitos o el análisis. Según OOHDM estas fases son idénticas a las que se deben realizar en la propuesta de OMT. Presenta otras

deficiencias, como que ha dejado fuera de su ámbito un aspecto esencial que es el tratamiento de la funcionalidad del sistema. El qué se puede hacer en el sistema y en qué momento de la navegación o de la interfaz se puede hacer, es algo que no trata y que lo deja como tarea de implementación. Además no ofrece ningún mecanismo para trabajar con múltiples actores. Por ejemplo, si la interfaz y la navegación de la aplicación varia sustancialmente dependiendo de quién se conecte a la aplicación, el diagrama navegacional, los contextos navegacionales y los ADVs resultarían muy complejos para representar esta variabilidad.

SOHDM tiene como ventaja que es un proceso sencillo de seguir, aunque se le puede criticar el hecho de que su nomenclatura está muy cerrada. Por ejemplo, para el desarrollo de la interfaz se define cómo se representa una imagen o un botón en el modelo, aunque no se dice nada de cómo se representa un elemento de audio, sin dejar ninguna opción a que el diseñador pudiese definir su propia representación.

HFPM ofrece las pautas que se deben seguir para el desarrollo de las aplicaciones multimedia. Sin embargo, no ofrece una metodología detallada. Sólo indica qué se debe hacer y no cómo se debe hacer. También puede observarse que se basa bastante en OOHDM y en los lenguajes de modelado orientados a objetos (OMT, UML, etc.). En definitiva HFPM integra las propuestas clásicas orientadas a objetos con la de OOHDM, reutiliza los modelos presentados por éstas y ofrece una directriz clara a seguir en el proceso de desarrollo.

OO/Pattern Approach no deja muy claro qué documentación hay que presentar y no hace ningún tipo de referencia a la interfaz.

RUP, y la extensión del lenguaje modelado UML, muestra análisis similares a las otras metodologías y no se especializa en una clasificación de producto, sino que generaliza a través del uso de la semántica original de UML OMMMA-L. Además unifica los mejores elementos de metodologías anteriores, siendo esto una ventaja irrefutable.

Debido a esto y a que en las comparaciones anteriores de acuerdo a los resultados la metodología RUP cumple con todos los aspectos señalados, el siguiente trabajo propone el uso correcto de la extensión de UML OMMMA-L en la Universidad de las Ciencias Informáticas, específicamente en los proyectos multimedia de la facultad 8.

3.2. Bases de la propuesta

La propuesta seleccionada anteriormente está avalada por la comparación entre las distintas metodologías y por los resultados de las entrevistas realizadas al personal calificado para ello.

3.2.1. Métodos, procedimientos y técnicas utilizados.

"El método científico es una especie de brújula en la que no se produce automáticamente el saber, pero que evita perdernos en el caos aparente de los fenómenos, aunque solo sea porque nos indica como no plantear los problemas y como no sucumbir en el embrujo de nuestros prejuicios predilectos."

El método independiente del objeto al que se aplique, tiene como objetivo solucionar problemas.

En la investigación se usaron los Métodos teóricos: Histórico lógico, Hipotético deductivo, como métodos de investigación.

Los métodos teóricos permiten comprender el fenómeno que se estudia, su evolución, elaborar la hipótesis y proponer las mejoras a los problemas que se identificaron.

Se analizó cómo han venido surgiendo y desarrollándose nuevos conceptos como el de Multimedia, Hipermedia e hipertexto y otros que han sido retomados como el de Metodología para el desarrollo de software. Las diferentes etapas por los que ha pasado la industria cubana del software y la influencia que ha tenido el desarrollo de las TIC en la sociedad y en la producción.

En la investigación se planteó todo el problema, donde los datos tomados de los proyectos multimedia de la facultad 8, la dinámica de los procesos realizados para desarrollo de éstos y la alta interrelación entre todos los factores que influyen, se funden como un todo en un sistema sostenible integral.

Se prestó sumo interés en los problemas de la producción de software multimedia en la facultad y un enfoque desde la utilización de metodologías que garanticen la planificación y control de los mismos. Desde la perspectiva histórico-lógica, en la primera parte de esta investigación se desarrolló un estudio del estado del arte del uso metodológico y de la experiencia internacional; se revisaron las ventajas y desventajas de cada una de las propuestas metodológicas y las tendencias a la resolución de esta problemática.

Es importante destacar que la revisión bibliográfica constituyó un método importante para el desarrollo del proceso de investigación y para apropiarse de los conocimientos relacionados y con el tema y necesarios para la garantizar la calidad del mismo.

La investigación sigue además un método hipotético deductivo porque a partir del problema concreto se plantearon objetivos específicos y la hipótesis que en el transcurso de la investigación son resueltas siguiendo métodos fundamentados.

Los métodos empíricos permiten describir y explicar las características del fenómeno en estudio.

Los métodos particulares dentro de los empíricos se aplicaron con el objetivo de recoger los datos necesarios para identificar el problema e identificar las causas que lo provocaron. El método particular utilizado en el trabajo es la entrevista.

Las entrevistas realizadas, fueron vitales para el desarrollo de la investigación, para avalar los conceptos que se manejan en la investigación y medir el alcance y la importancia que tiene la temática. Captar la información cualitativa y cuantitativa del fenómeno, conocer los criterios sobre la forma en que se organiza y se lleva a cabo la producción de software multimedia en la UCI, así como las posibles soluciones que se proponen en la investigación. Para ello se

entrevistaron personas involucradas en la producción de software multimedia en la UCI y fuera de ella, con cierto grado científico.

Entrevista tipo 1

La población a estudiar fueron especialistas en producción de software multimedia externos a la UCI. La selección se realizó con la técnica de muestreo no probabilística, muestreo intencional. Se seleccionaron personas con grado científico y que participan en eventos nacionales e internacionales que han estado vinculados a la universidad o a la industria cubana del software.

Se seleccionaron 3 especialistas vinculados a la producción de software multimedia y a las metodologías para el desarrollo de este tipo de software en la empresa CITMATEL, y 1 especialista en el Instituto Superior Politécnica José Antonio Echeverría, donde se desarrolla este tipo de software. Todo esto con el objetivo de saber el estado de usabilidad en que se encuentran las metodologías cubanas en el país.

Entrevista tipo 2

La población a estudiar fue el personal involucrado en los proyectos de producción de software multimedia de la facultad 8 de la UCI. La selección se realizó con la técnica de muestreo probabilística, muestreo intencional para poder obtener la mayor representatividad e información posible, de acuerdo con los intereses de la investigación, la cual fue entrevistar a personas que vinculadas a la producción software multimedia y que por ende han tenido que enfrentar diferentes problemas que le han permitido ir desarrollándose.

Se seleccionaron 2 directivos del departamento de Ingeniería de Software, y el vicedecano de producción de la facultad, con el objetivo de saber el principal problema de los proyectos y saber además cuál era el conocimiento a cerca de las metodologías de desarrollo de software multimedia en la facultad.

De los 7 proyectos multimedia de la facultad se entrevistaron 5 Líderes y 30 desarrolladores de los mismos, teniendo en cuenta la cantidad de proyectos de la facultad la muestra para la investigación es del 100%. El objetivo esta vez era identificar la metodología usada en el

proyecto, el procedimiento de selección de la misma, en caso negativo identificar las causas fundamentales por la cual no usan metodologías y los problemas que trae no usarla. Saber el grado de conocimiento de los involucrados en el tema de las metodologías multimedia.

Los resultados obtenidos de estas entrevistas fueron los siguientes:

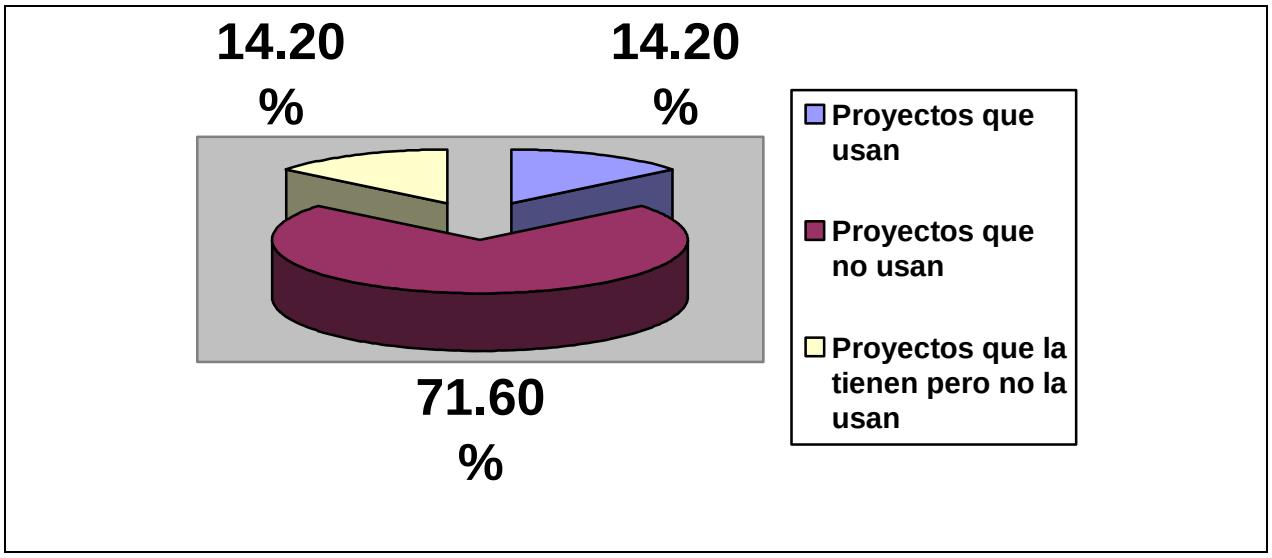


Figura 10. Gráfico de usabilidad de las metodologías en los proyectos

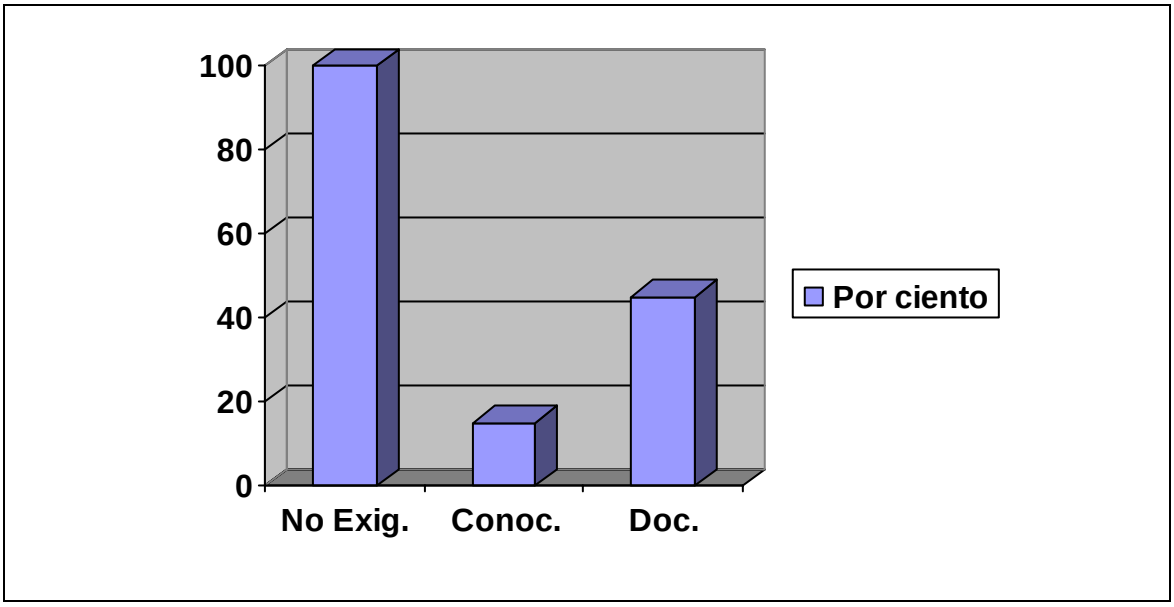


Figura 11. Gráfico de las causas de la no usabilidad de las metodologías

Lo anterior se realizó con el objetivo recopilar elementos a tener en cuenta para la solución y la validación de la propuesta. Todo esto reveló que 6 de los 7 proyectos no usan metodologías. Aunque uno de estos proyectos tiene todo un análisis de ingeniería, pero sólo documentado y archivado. Las principales causas de este fenómeno es la no exigencia por parte de la dirección de la facultad, la falta de documentación de las distintas metodologías existentes, lo que conlleva al desconocimiento de las mismas.

En los Anexos # 4 y 5 se pueden ver el diseño los modelos de la entrevista aplicada a los especialistas en la producción de software y en el uso de las metodologías multimedia.

3.2.1.1. Resultados del uso de la metodología propuesta en el proyecto “Historia Universal”

Las investigaciones realizadas revelaron que solo uno de los proyectos entrevistados usa metodología. Este proyecto es el “Historia Universal”, un proyecto en conjunto con la Universidad de la Habana, destinado a mostrar a través de una multimedia todo lo concerniente a la historia. El proyecto en sus inicios no tenía ningún análisis metodológico, pero mediados se integró el lenguaje de modelado OMMMA-L, extensión de UML de RUP, por lo que tuvieron que adaptar los métodos anteriores a los establecidos por el mismo y otros sí tuvieron que crear nuevos.

Se optó por OMMMA – L ya que permitió:

- Modelar la estructura a través de diagramas de objetos y clases.
- Describir el comportamiento a través de los diagramas de interacción.
- Realizar la distribución espacial de media contemplada en el modelo vista a través de la descripción de los diagrama de presentación. (Artefacto nuevo propuesto por OMMMA-L. La semántica asociada a dichos diagramas, conservan en muchos casos su significado, en otras se adaptan a la interpretación de los conceptos propios de multimedia.)

El tiempo de ejecución si se ha optimizado ya que las ideas de qué diseñar e implementar se modelan de forma más eficiente y comprensible. Actualmente el proyecto esta en fecha de entrega aunque aún se trabaja en algunos detalles.

Todo esto gracias a que OMMMA-L no es un lenguaje nuevo sino una extensión del UML que se imparte en la universidad, por lo que no es necesario aprenderlo, sino interpretar las características extendidas, centrados a la lógica de funcionamiento de una multimedia, que es por lo general, sencilla.

OMMMA-L es una extensión de UML por lo que tienen muchas cosas en común solo lo diferencia los artefactos que este incorpora en cada uno de las fases de desarrollo.

Las ventajas de OMMMA-L van encaminadas a mostrar un mejor modelado de estos productos multimedia desarrollados en ambientes orientados a objetos.

El éxito alcanzado en la realización de este proyecto se debe a la utilización de la metodología correcta, que indicó cómo trabajar eficientemente e cada una de las fases de desarrollo evitando caer en las deficiencias de los otros proyectos multimedia de la facultad 8. Aumentó la calidad del mismo a partir de su utilización por medio de una mayor transparencia y control sobre el proceso; “producir lo esperado en el tiempo esperado y con el coste esperado” aunque esto último se obvia dado que es una colaboración entre ambas universidades.

Conclusiones

Basado en las especificidades de los proyectos de desarrollo de software multimedia de la facultad 8 y en el análisis comparativo de las distintas metodologías de desarrollo, en este capítulo se sentaron las bases y se propuso la metodología más adecuada a usar en la universidad, específicamente en la facultad de la especialidad.

CONCLUSIONES

En el presente trabajo se realizó un análisis del estado del arte sobre diferentes metodologías para el desarrollo de software multimedia existentes en el mundo y en Cuba. Se realizó una comparación entre todas éstas basados en tres aspectos fundamentales. Se realizaron una serie de entrevistas, las que revelaron la situación actual de los proyectos de desarrollo de software multimedia de la facultad 8. Basados en estos resultados y del análisis comparativo se obtuvo la metodología mas completa, siendo ésta la propuesta a usar en los proyectos dedicados al desarrollo de multimedia de la universidad, específicamente en los proyectos de la facultad 8. Con esto se dio cumplimiento a los objetivos del trabajo.

El uso de esta propuesta aumentará la calidad de los proyectos multimedia de la facultad 8. Pues la relación tiempo-costo-calidad se vera favorecida, por las ventajas que permite usar una metodología de desarrollo de software.

RECOMENDACIONES

En la facultad 8 se deberá impartir cursos optativos sobre las distintas metodologías de desarrollo de software y de la extensión del lenguaje de modelado UML, OMMMA-L para multimedia con el objetivo de elevar el nivel de conocimiento de los estudiantes.

Se deberá actualizar el sitio http://dirproduccion2.uci.cu con las restantes metodologías para desarrollo de productos multimedia.

Se deberá profundizar en las nuevas metodologías de entorno multimedia que están naciendo.

REFERENCIAS BIBLIOGRÁFICAS

FRANCA GARZOTO, D. S., PAOLO PAOLINI. *HDM-A Model Based Approach to Hypermedia Application Design.* ACM Transactions on Information System, 1993. p.

Fundamentos de ingeniería de software. Disponible en: http://148.202.148.5/cursos/cc321/fundamentos/unidad7/tema7_1.html

GRADY BOOCH, J. R., IVAR JACOBSON. *Unified Modeling Language User Guide*. Ed. Addison-Wesley, 1999. p.

GUSTAVO ROSSI, D. S., ALEJANDRA GARRIDO. *Design Reuse in Hipermedia Applications Development*. Southampton, Proceedings of ACM International Conference on Hypertext 1997. p.

HEESEOK LEE, C. L., CHEONSOO YOO. *A Scenario-based object-oriented methodology for developing hypermedia information systems*. Processing of 31st Annual Conference on Systems Science. , 1998. p.

IVAR JACOBSON, G. B., JAMES RUMBAUGH. *The Unified Software Development Process*. Ed. Addison-Wesley, 1999. p.

J. THOMSON, J. G., J. COOKE. *Algorithmically detectable design patterns for hypermedia collections*. 1998. p.

LANGE, D. B. *An Object-Oriented Design Approach for Developing Hipermedia Information Systems*. Japón, IBM Research, 1995. p.

OLSINA, L. *Building a Web-based information system applying the hypermedia flexible process modeling strategy*. 1st International workshop on Hypermedia Development, 1998. p.

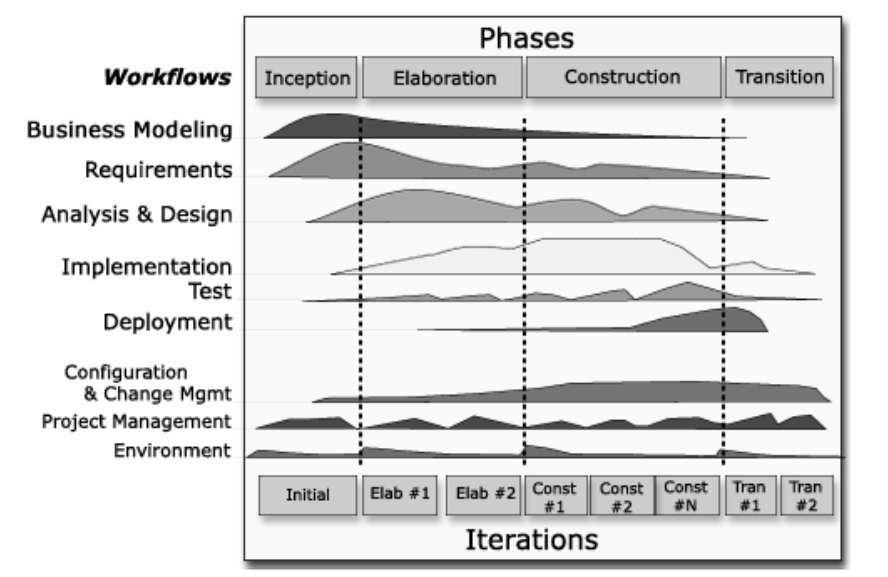
REBECCA WIRFS-BROCK, B. W., LAUREN WIENER. *Designing Object-oriented Software*. Prentice-Hall 1990. p.

RUMBAUGH, J. *Modelado y Diseño Orientado a Objetos*. Ed. Prentice Hall, 1991. p. .

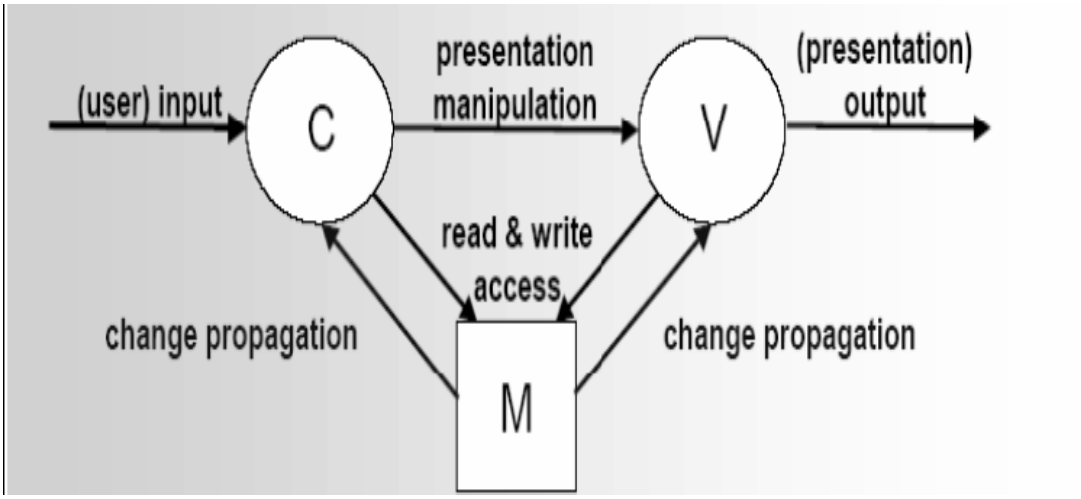
TOMAS IZAKOWITZ, E. S., P. BALASUBRAMANIAM. *RMM:A methodology for structured hypermedia design*. ACM Press, 1995. p.

YOURDON, E. *Modern Structured Analysis*. Estados Unidos, Prentice-Hall, 1989. p.

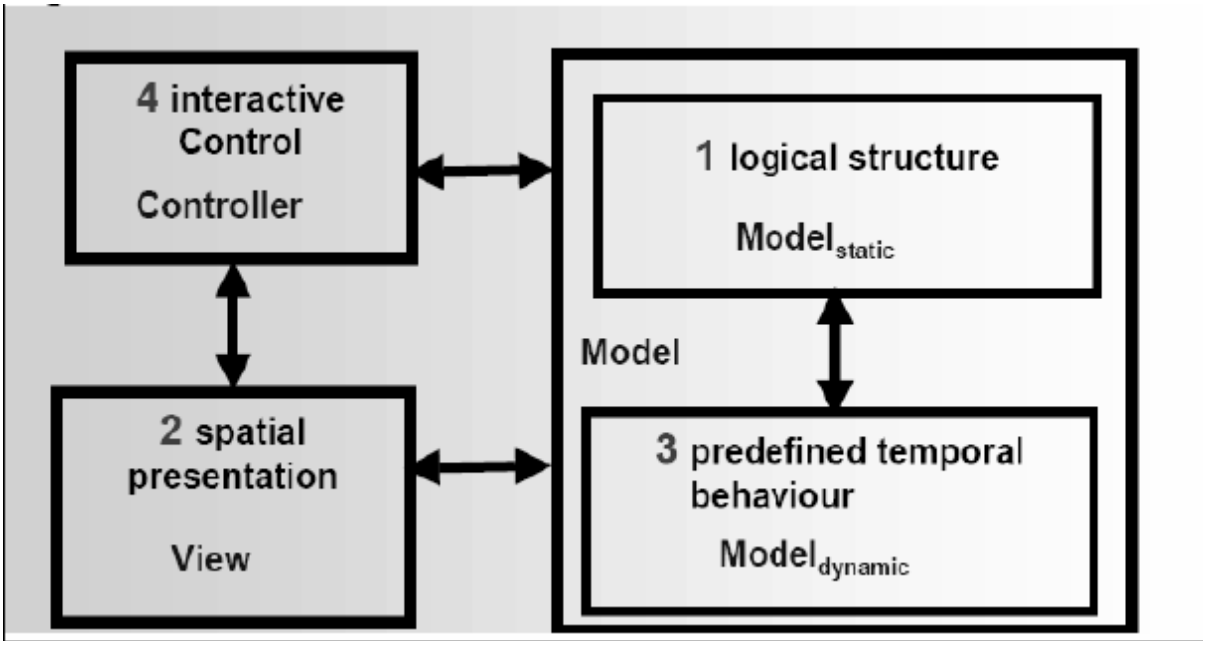
ANEXOS



Anexo 1. Fases e iteraciones de la metodología RUP



Anexo 2. Patrón Modelo Vista Controlador



Anexo 3. Patrón Modelo Vista Controlador para multimedia

Anexo 4. Diseño de la entrevista de tipo 1

1. ¿Qué metodología usan en el proyecto?

En caso de responder positivamente, diga:

a- ¿La comenzaron a utilizar desde el principio?

2. ¿Cómo eligieron la metodología que están usando actualmente?

En caso de responder negativamente…

1. ¿Qué problemas trae no usar una metodología en el desarrollo del proyecto?
2. ¿Qué metodologías conocen?

Anexo 5. Diseño de la entrevista de tipo 2

Entrevista realizada a los especialistas de la empresa CITMATEL y en la CUJAE

1. ¿Bajo que condiciones fue elaborada la metodología?
2. ¿Cuáles son las ventajas que trae usar una metodología en el proceso de producción de software?

3. ¿Cuáles son las metodologías para desarrollo de software multimedia que conoce?
4. ¿Cuáles son las que más se usan en el país?
5. ¿Actualmente se usa la metodología en el centro?