Aplicaciones Educativas Multimedia , el Problema de la Evolución del Software.

Manuel Pérez Cota, Francisco J. Vázquez Núñez, Amparo Rodríguez Damián,
Jacinto González Dacosta
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Vigo
Lagoas de Marcosende s/n. Apdo. 874 36200 - Vigo España
Tel: +34-(9)86-812662Fax: +34-(9)86-812661
e.mail: mpcota@uvigo.es www: http://www.lsi.uvigo.es/lsi/mpcota/mpcota.html

Resumen:

En las aplicaciones multimedia, el usuario se enfrenta al gran problema de decidir qué camino seguir para conseguir un adecuado diseño de sus aplicaciones y, en ese camino, lograr no perderse en una gran cantidad de metodologías, estilos de programación, programas de diseño, de autor, etc. Si a esto le añadímos el decidir si una aplicación funcionará o no conectada a Internet, si la aplicación utilizará o no unos u otros tipos de imágenes o sonidos, convierte lo que puede parecer un sencillo trabajo en papel en un complicado desarrollo que, tanto los informáticos como los no informáticos afrontan en un compromiso de trabajo en equipo que acelera el proceso y hace que se consigan en muchas ocasiones resultados espectaculares, pero que obligan a pensar en el tipo de hardware y software a utilizar.

El autor de software es el alma mater de cualquier desarrollo, pero además se convierte en el creador, mezclador, modificador e incluso el probador para que todas estas cosas acaben funcionando bien (que generalmente lo hacen pero a costa de un excesivo esfuerzo). Es necesario ordenar lo que se quiere hacer mezclando diversas etapas de desarrollo, que en los sistemas antiguos se denominaban etapas de análisis y diseño y posteriormente Ingeniería de Software. Hoy día la ingeniería de software, aplicada al desarrollo multimedia, aporta al usuario muchos conocimientos que pueden ayudarle a simplificar su trabajo pero por ser tantos pueden hacerle difícil tomar una decisión, no obstante con una correcta organización y un mejor conocimiento de los recursos de que dispone puede mejorar el estilo de trabajo, aquí intentamos

Palabras clave: Multimedia, Hipermedia, Hipertexto, Metodologías y Herramientas de desarrollo, Orientación a Objetos, Interactividad.

1.Introdución.

Como todo en la informática, la creación de sistemas informáticos multimedia se encuentra en constante y rápida evolución. Muchos profesionales estudiamos, día a día, la forma de conseguir productos que permitan que el desarrollar un sistema multimedia sea cada vez más sencillo y más cercano a cualquier persona, dado que hoy en día son más los no profesionales de la informática los que, junto con los profesionales o de forma libre, intentan crear aplicaciones multimedia de mejor calidad que les resuelvan sus problemas.

Sin embargo, no son sólo los autores no profesionales los que intentan realizar nuevos trabajos, todos los profesionales de la informática están en ello, pero la cantidad de nuevas herramientas de desarrollo, los nuevos métodos y metodologías, los nuevos

lenguajes, el nuevo hardware, etc., hacen que la labor de desarrollo se torne cada día en algo más complejo y sutil.

La tendencia actual a trabajar con prototipos, junto con la aparición de herramientas de red que permiten al usuario "estar" donde se produce la tecnología permite pensar en un acercamiento del usuario a la tecnología y un alejamiento del mismo usuario de su capacidad de absorber todo lo que aparece en el mercado.

En 1991, Graham [grah91] hacía el siguiente comentario sobre los métodos orientados a objetos respecto de los métodos clásicos de programación: "...se están empezando a utilizar nuevas metodologías de modelización y desarrollo. De éstas, una de las más importantes es la orientada a objetos, que se basa en una interpretación de la realidad a partir de entidades (objetos) que tienen propiedades (atributos) y funciones (métodos). La gran variedad de áreas de aplicación, junto a las ventajas que ofrece frente a las metodologías convencionales, permite prever un gran impacto durante los próximos años en las tecnologías de la información"; este mismo pensamiento podemos aplicarlo hoy en día, pero a la ventaja que ofrece Internet y la utilización masiva de redes para el trabajo cooperativo o en grupo, con la desventaja que suponía el trabajo independiente o en entornos muy poco cooperativos.

2. La evolución en el Desarrollo de Proyectos Informáticos.

Como se puede observar en la introducción, el panorama que se presenta a los autores es de una gran complicación, pues la realidad dice que tienes múltiples, casi podríamos decir infinitos caminos para desarrollar un proyecto informático, a continuación formularemos algunos de estos caminos de forma que podamos después dar una visión de lo que nosotros opinamos que se puede hacer para no perderse en el maremagnum de fórmulas de trabajo.

Comencemos hablando de los ciclos de vida de los proyectos, desde los clásicos hasta los más recientes.

Por definición el ciclo de vida de un proyecto podría definirse como "el conjunto de etapas que permiten resolver un problema". Si estamos hablando de un proyecto informático la definición podría quedar como "el conjunto de etapas que permiten crear una solución que siendo utilizada por el (los) computador (es) resuelve un problema"; en esta definición aparece claramente un nuevo problema, y éste es que las distintas fases que conforman la solución no pueden ser consideradas independientes del análisis y diseño del conjunto del sistema en que va a integrarse, es decir la conjunción del hardware y el software.

En este contexto, la Ingeniería Informática, juega un papel relevante como disciplina que ha de llevar a cabo el diseño del sistema y que resolverá las necesidades de los usuarios y sus requerimientos operacionales. Añadiendo a esto los componentes de cualquier proyecto de ingeniería como son: seguridad, fiabilidad, mantenimiento, disponibilidad, completitud, etc. [kell92].

Analicemos las fases clásicas de un proyecto utilizando ingeniería del software, y su explicación [pres97], entrando en lo que afectan al autor de desarrollos multimedia educativos. En primer lugar el Análisis de requerimientos, que nos daba una visión en conjunto del proyecto a realizar, en este punto se realizaba un estudio de las necesidades "operacionales" de los usuarios y se determinaban las distintas restricciones con las que tendría que trabajar el sistema y se definían las funciones que debería realizar. En el caso de los sistemas educativos esto podría ser muy sencillo si tuviéramos perfectamente acotado el campo de desarrollo y los sistemas sobre los que se deseaba trabajar, que en muchos casos era lo normal. Este entorno ha variado muy

poco desde entonces, pues se suele seguir teniendo acotado tanto un tema como el otro, es decir las restricciones se encuentran claramente.

La segunda fase que nos traía el análisis de requerimientos es la de los requerimientos del software. En este caso nos referíamos a la forma en que dividiríamos nuestro problema para abordarlo adecuadamente, lo normal era decidir sobre las conexiones o relaciones con los operadores o usuarios, las restricciones que nos imponían los recursos a utilizar y tomar la decisión de lo que se entregaría en cada fase.

En los sistemas educativos aparecían en este punto los problemas del conocimiento de los usuarios finales y su capacidad para utilizar uno u otro tipo de hardware, convirtiéndose el autor en una persona que debería –si cabe- acotar claramente las líneas de actuación del proyecto definiendo de forma genérica los materiales que pretendía fueran utilizados.

En la siguiente fase nos encontramos con el diseño del software, en el cual se establecían los distintos componentes que hacían que los requerimientos quedaran satisfechos. Generalmente se establecía un diseño general, estableciendo los distintos módulos a construir y el diseño detallado en el que se establecían, o mejor dicho, diseñaban o desarrollaban los algoritmos a utilizar y las estructuras de datos de cada módulo.

En esta etapa, generalmente el autor o los autores, trabajaba pensando en un lenguaje de programación concreto, generalmente el que le permitía más libertad y menos complejidad en su trabajo. Cabe recordar aquí el gran éxito que desde un principio tuvo el lenguaje BASIC en el desarrollo de aplicaciones educativas. Uno de los primeros tutoriales conocidos, el tutorial de BASIC de Hewlett-Packard [hewl79] estaba integramente desarrollado en BASIC, desbancando al FORTRAN, al RPG y al COBOL en esos menesteres en los 70 y principios de los 80 [pere79].

En la siguiente etapa, es decir, la implementación o codificación se escribían los algoritmos obtenidos en la fase anterior en un lenguaje de programación. En el párrafo anterior se da una idea de cómo los autores realizaban esta función.

En la fase de pruebas, o también tests, se procuraban y procuran la detección de errores, dos tipos básicos: las pruebas de integración (Pruebas alfa o Alfa tests) y las pruebas de campo (Pruebas beta o Beta tests). En las primeras los propios autores prueban por partes e integrado su

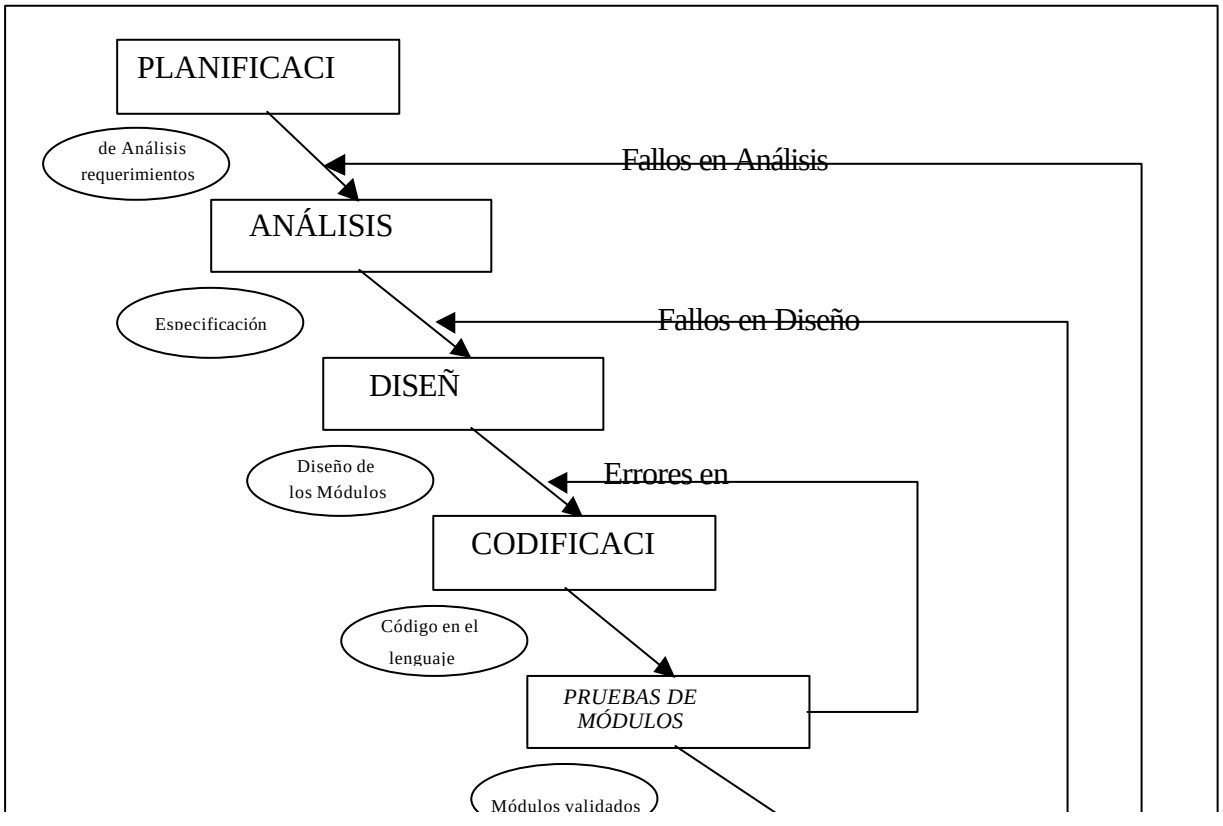

Figura 1: Ciclo de vida clásico dinámico

sistema completo, en las segundas generalmente se envía a una muestra escogida de usuarios el paquete a probar y se le pide que realice las pruebas, se puede hacer con y sin documentación y formación, obteniéndose como resultado las correcciones que el proyecto pueda necesitar.

En esta etapa, el software educativo era probado por los propios diseñadores y generalmente probado por colegas quienes hablaban de las bondades o maldades del diseño realizado.

La última etapa, el mantenimiento permitía y permite al autor del software re-diseñar su propuesta y adaptar su diseño original a los nuevos tiempos que pueden aparecer, pero que como se dijo en la introducción significan no sólo el cambio en la utilización del software sino también la utilización de nuevo hardware, y lo que puede modificar más un proyecto, la aparición de interconexiones con otros equipos, nuevas formas gráficas, hipertextos, hipermedias (aunque el hipertexto sea un subconjunto de la hipermedia, lo separamos por claridad conceptual) etc..

Algo a tener en consideración en este tipo de metodologías era que su rigidez acabó llevándolas a caminos en los cuales las etapas que podríamos llamar de pruebas se realizaban en todas las etapas, apareciendo el ciclo de vida clásico estático o ideal y el dinámico o realista, éste último tenía continuos retornos. Esto en general no era malo pues se corregían muchos problemas sobre la marcha, el problema surgió cuando esas correcciones, o mejor, faltas de previsión llevaron a que se comenzaran a hacer los proyectos demasiado largos en el tiempo, superando prácticamente siempre las previsiones originales y que además los costes se dispararan. En las figuras 1 y 2 puede verse en forma gráfica lo aquí dicho.

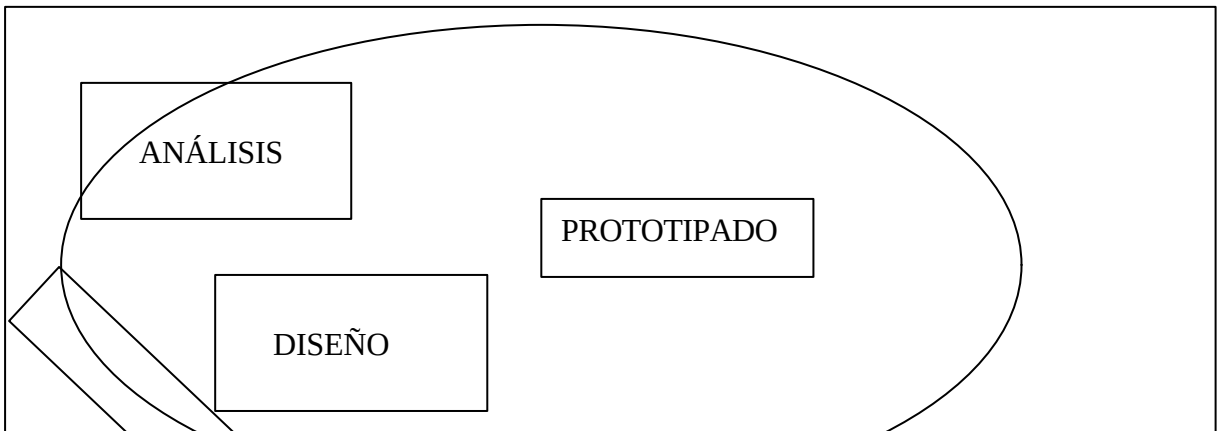

Figura 2: El prototipado y la proporción de costes

A las metodologías clásicas se añadió en una etapa posterior una nueva fase que llegó precedida de una serie de herramientas que aparecieron con el fin de ayudar a crear entornos que podían ser similares al entorno final (nos referimos a las pantallas de presentación, formularios, salidas impresas, etc.) a esta etapa se llamó prototipado. El prototipo es como una versión recortada del proyecto final, pero que en las distintas etapas iniciales puede dar una idea a los autores (tanto a los diseñadores como a quienes proponen) de como pueden quedar las cosas y la mejor forma de visualizar algo. También aquí tenemos dos formas clásicas de prototipos, éstas son: los horizontales o experimentales y los verticales o de exploración; los primeros permiten ver todas las funciones del sistema pero no totalmente desarrolladas, de forma que los usuarios pueden imaginarse cómo será el sistema una vez que esté terminado. En el segundo las partes visuales del sistema están prácticamente desarrolladas, e incorporan sus facetas finales, no obstante el producto no funciona y sólo sirve para –explorar- y saber si se está de acuerdo con lo que sería el resultado visual final.

El prototipo va, como lo mostramos en la figura 2, insertado dentro de las etapas de análisis y diseño, y utilizando parte de la etapa de codificación.

3. La orientación a Objetos.

La constante evolución de la informática ha llevado a la modificación de muchos métodos, de muchos lenguajes y de la forma de trabajar con el computador. Se ha pasado de utilizar pantallas tontas, en las que solamente era posible visualizar texto, a sistemas inteligentes con capacidades gráficas de muy alta resolución y que además tienen sonido claro.

En este orden de cosas el desarrollo de los proyectos informáticos ha cambiado modificando la forma en que se realizan el análisis, el diseño, la codificación, las

pruebas e incluso el mantenimiento. Hemos pasado de utilizar metodologías estructuradas a modelos de datos y de éstos a las metodologías orientadas a datos. En ellas el esquema conceptual se basa en un conjunto de entidades, en los atributos que las describen y las relaciones existentes entre ellas [chal97], este modelo entidad/relación se encuentra muy utilizado en proyectos de bases de datos.

Frente a estos métodos, el clásico y el orientado a procedimientos de entidad/relación, surge una nueva forma de enfocar el problema en el que se busca permitir y facilitar la reusabilidad, el refinamiento, la verificación y la ampliación del software sin crear grandes problemas añadidos. Conseguir esto obliga a romper con viejos esquemas en los que un dato puede ser válido en todo un sistemas independientemente del número de funciones que lo utilicen, obliga a romper con el conocimiento total de lo que ocurre dentro de cada módulo, obliga a reutilizar procesos bien diseñados, es decir nos obliga a diseñar basándonos en principios de ocultación de información, herencia y polimorfismo.

Todo esto nos hace cambiar la forma de pensar y de trabajar en un proyecto informático, ahora intentamos que un algo pueda ser convertido en otro algo informático, no buscamos su código, sino su todo pero ejecutado por un computador.

El modelo orientado a objetos es en sí más natural que el modelo clásico, pues como indica Booch [booc94] referido a Robson [booc94] ...”el modelo de objetos resulta atractivo para el funcionamiento de la cognición humana, porque, como sugiere Robson, -muchas personas que no tienen ni idea de cómo funciona un computador encuentran bastante natural la idea de los sistemas orientados a objetos-“.

El principal cambio, respecto a los anteriores modelos de trabajo, resulta de la modificación de la parte de análisis pues las operaciones de los objetos se encuentran ocultas dentro de ellos y no son generales como en los modelos anteriores.

La tecnología orientada o objetos ha provocado un profundo cambio en el sentir y en el modo de trabajar de los informáticos, han aparecido metodologías nuevas Coad, Rose, etc. Metodologías en las que el proceso clásico se convierte en un doble proceso (macroproceso y microproceso). En el primero se establecen un orden que consiste, según Booch(9) en:

Establecimiento de los requisitos centrales (hacer el modelo conceptual)

Desarrollo de un modelo del comportamiento deseado (análisis)

Creación de una arquitectura (diseño)

Transformación de la implementación (evolución)

Gestión de la evolución posterior a la entrega (mantenimiento)

En el segundo se establece:

Identificación de las clases y objetos a un nivel dado de abstracción.

Identificación de la semántica de estas clases y objetos.

Identificación de las relaciones entre estas clases y objetos.

Especificación de la interfase y después la implementación de las clases y objetos.

Esto, como se puede ver, desarrolla el modelo clásico pero implementándolo para los objetos. En la figura 3 podemos ver algunas de las cosas que rodean el modelo orientado a objetos, y observar como este modelo es el que ha revolucionado el trabajo del informático, pero en ningún caso lo ha simplificado, en el sentido de dejar todo en una sola metodología o en una sola forma de trabajo, mas bien al contrario, como lo decíamos párrafos atrás, ha complicando un poco mas la vida a quien se dedica a la informática.

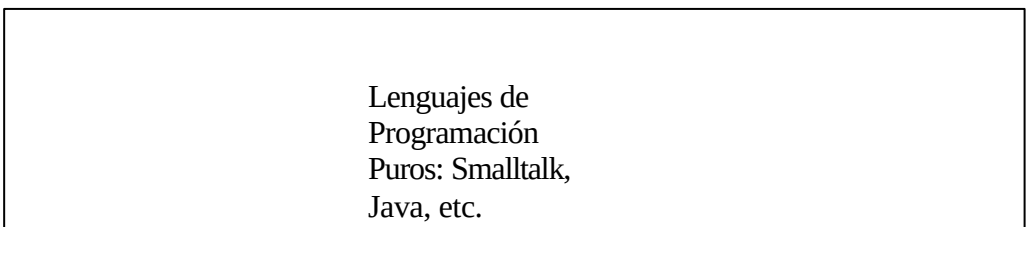

Figura 3: Consecuencias de la orientación a objetos

4. Aplicaciones Multimedia, Hipermedia e Hipertexto.

Con la orientación a objetos y todos sus derivados fueron apareciendo en el mercado nuevos sistemas que simplificaban la relación usuario/computador, pero esa simplificación de la relación es debida a que el computador se acerca a su forma de trabajar al lenguaje objetual y pierde el lenguaje de comandos; el usuario se encuentra más a gusto trabajando con la máquina. Con la aparición a gran escala del acceso a Internet, el usuario no sólo trabaja con su máquina sino que puede acceder a todo un mundo de posibilidades que además pueden estar en cualquier parte de la red.

Esto que es positivo para el usuario, no lo es tanto para el desarrollador es decir el autor de software, quien de golpe necesita que sus sistemas cumplan muchas más condiciones de las que sólo unos años atrás se pensaba como si fueran condiciones de ciencia ficción.

Estas nuevas condiciones o condicionantes, que también serán fuertes restricciones, son en un orden de aparición:

1)Los gráficos de alta definición.

2)El sonido.

3)Los gráficos en movimiento o escenas.

4)La conexión hipertextual.

5)La conexión hipermedial (recuerdese el hipertexto como subconjunto hipermedia).

6)La multimedia (entendida como utilización de múltiples recursos paralelos).

En el primer punto, la única complicación que encuentra el autor es su capacidad para hacer que en el entorno que está creando aparezcan los gráficos. Hoy día este tema se encuentra totalmente resuelto con casi cualquiera de los sistemas de diseño orientados a objetos, lo mismo podemos decir de los propios lenguajes orientados a objetos, que tienen clases que resuelven sin mayor problema el tema de la aparición de gráficos. La complicación reside, como veremos más adelante, en el análisis y diseño del sistema en sí.

Al igual que los gráficos, el sonido es un tema que los diseñadores de los lenguajes han resuelto bastante bien. Existen en casi todos los lenguajes orientados a objetos clases que solucionan el problema de la generación o captura de sonidos sin mayores complicaciones, pero al igual que en el punto anterior el problema aparece a la hora del análisis y diseño del sistema.

Los gráficos en movimiento, o como llaman algunos autores, las escenas, complican un poco más la situación pues aquí sí que ya son pocos los lenguajes que permiten la utilización de clases específicamente diseñadas para escenas, a no ser los lenguajes (sistemas) de complementación de autor (authorware, hypercard, etc.), que dan al usuario la posibilidad de "cargar" secuencias que dan la sensación de acción, generalmente aquí es el propio usuario el que se las debe apañar para crear ficheros y objetos que capturando esos ficheros den la sensación de acción.

La conexión hipertextual, es un concepto que, aunque no es nuevo, aparece muy ligado a las páginas web, de hecho, muchos usuarios saben lo que es el hipertexto gracias a las páginas de Internet, pero no se imaginaban que eso era lo mismo que lo que algunos programas llamaban índices enlazados por palabras. En esta fase el autor debe tener muy claro qué es lo que quiere enlazar y cómo. Para esto lenguajes como el HTML, ayuda mucho pues da al autor más novel la posibilidad de realizar enlaces con cierta facilidad, pero al igual que en los dos puntos anteriores esto complica mucho las fases de análisis y diseño.

La conexión hipermedial, al igual que la anterior se basa en enlaces entre objetos, pero éstos no están basados sólo en texto, también lo están dependiendo de imágenes o cualquier otra cosa que permita un enlace dentro de un programa. Seguimos complicando el proceso.

Por último, debemos añadir a todo esto la aparición de más recursos a disposición del computador y del usuario del computador, que hace que se junten a éstos más piezas del rompecabezas, o mejor dicho, no sólo hace más piezas del rompecabezas, sino que las hace más pequeñas y con más posibilidades de acoplamiento.

Actualmente, el interés de la utilización de la hipermediación, se ha centrado en el uso de modelos y métodos de concepción de aplicaciones bien estructurados. El objetivo principal de los grupos que trabajan en estos modelos y métodos es el proponer aproximaciones de concepción de aplicaciones multimedia según las reglas del arte de programación orientada a objetos, es decir : la concepción bien documentada, el diseño de aplicaciones sencillas, fáciles de mantener y reutilizables.

Generalmente se dice que existen tres pasos para dotar a los sistemas multimedia de metáforas hipermedia:

1)Identificar y definir los elementos del dominio de la aplicación.

2)Identificar y definir los elementos hipermedia.

3)Identificar y definir la relación que les une.

Diversos autores ([alpi93],[blum96],[diaz96],[pasc95],[pere97-1],[vaug96]) han propuesto distintos modelos a seguir para conseguir que una aplicación pueda funcionar

como aplicación multimedia con el paradigma de hipermedia, y se puede decir que todos consideran que la aplicación puede ser todo componente (objeto, entidad, relación, atributo, evento, operación, abstracción,...) especificado en el curso del ciclo de vida de la aplicación, y por elemento hipermedia todo componente (nodo, enlace, ancla, método, evento,...) propio del modelo hipermedia. No implicando la realización el respeto a este orden dado. Es decir, que se acabe con un elemento no quiere decir que se comience con el siguiente.

En este sentido se han desarrollado modelos que intentan acercar la concepción hipermedia a los modelos clásicos entidad relación, como el HDM (Hipertext Design Model) que aporta la utilización de accesos a estructuras como son los índices y las visitas guiadas. Asimismo la metodología RMM (Relationship Management Methodology) que se basa en el diseño de aplicaciones hipermedia que han de sufrir constantes cambios pero cuya estructura es estable.

Otras metodologías trabajan sobre todo el ciclo de vida del desarrollo multimedia hipermedializado como la Metodología de desarrollo y diseño de multimedia [blum96] que claramente se basa en métodos pensados en educación, así en su parte de diseño tiene un diseño educativo y un diseño interativo. O la metodología MAJA [pere97-1], en la que siguiendo un ciclo de vida clásico orientado a objetos y dinámico, se busca que el autor pueda seguir todas las fases del ciclo de vida como si estuviera escribiendo una película, de forma que su objetivo final sea el obtener un producto funcional, reutilizable, documentado y fácilmente modificable .

5. Los Retos del Autor de Software.

Hasta aquí hemos seguido, con el fin de dar una visión de lo que ocurre a los diversos autores, el proceso de desarrollo de software en sus vertientes más sencillas y más duras, intentando plasmar lo que el autor se puede encontrar cuando intenta empezar a desarrollar un paquete de software

Como se puede deducir, con cierta facilidad, los problemas que sufre el actual autor de programas pueden ser los siguientes:

1)Exceso de metodologías de desarrollo.

2)Exceso de lenguajes de programación.

3)Hardware cada día más complejo.

4)Obsolescencia muy rápida de hardware y software.

A su favor se encuentra con:

1)Sistemas operativos más sencillos y flexibles

2)Mayor capacidad de búsqueda y obtención de información

3)Herramientas más baratas y fáciles de conseguir.

Es pues, a nuestro entender, el primer reto del autor el ser capaz de adaptarse, sin desanimarse, a las nuevas tecnologías hardware y software, de forma que los productos que diseña puedan cubrir siempre las necesidades para las que van diseñados. Esto generalmente se consigue leyendo. La mejor forma de mantenerse al día es leer, pero el reto consiste no en leer todo lo que caiga en las manos, sino en leer lo adecuado. Quizá el problema radica en saber qué leer, por lo general las revistas informáticas de gran calidad sean las que mejor cubren este primer reto, pues en ellas el autor puede encontrar una información actual y generalmente contrastada con expertos. El conseguir información por Internet, salvo en direcciones muy prestigiosas, suele no ser adecuado pues son pocas las páginas que se actualizan lo suficientemente rápido y desgraciadamente en algunas que se actualizan con cierta asiduidad los datos no se

encuentran contrastados, lo que en informática constituye un gran riesgo para el autor, sobre todo el que quiere empezar.

El segundo reto consiste en entender el significado de una metodología, con esto queremos decir, que el autor debe tener claro qué es lo que quiere hacer, en el caso que nos lleva este papel, nos referimos a las aplicaciones educativas, el autor de aplicaciones educativas debe pensar en metodologías orientadas hacia su trabajo. El escoger una metodología pensada para sistemas administrativos, sistemas en tiempo real, etc., le llevarán a encontrarse que en ciertas etapas tiene que desarrollar puntos que no tienen mayor interés y sin embargo partes que pueden ser transcendentes para su trabajo quedar muy descubiertas o simplemente enredarse de formas que son muy difíciles de resolver. El reto aquí consiste en buscar una metodología que cubra adecuadamente el trabajo que se está realizando, pensando que esa ha de ser la metodología que se utilice para todos los trabajos, de esta forma estandarizará sus aplicaciones.

Parte de lo anterior cubre el tercer reto que es el de entender la metodología que se está utilizando, no vale el utilizar en un paso un método y en otro paso otro, si se admite una metodología ésta se debe admitir para todo el curso del trabajo y respetarla en toda su extensión. Es típico en los informáticos [pres97] el comenzar utilizando una metodología y después sin una razón aparente saltársela y pasar a otra o simplemente utilizar únicamente parte de ella. Esto obviamente es un error que normalmente se paga muy caro. Afortunadamente la mayoría de las herramientas de desarrollo para ciertas metodologías actuales no permiten en ningún modo que el usuario se salte ningún paso, lo que puede permitir obtener sistemas robustos y fácilmente reutilizables y mantenibles.

El cuarto reto consiste en acertar con el lenguaje de programación adecuado. Si uno lee los artículos de las revistas especializadas se pueden leer comentarios favorables y desfavorables de todos los lenguajes de programación. No obstante, si uno se para a ver cuales son los lenguajes motor de muchas de las aplicaciones que en el entorno educativo se realizan, hoy por hoy nos encontramos con que el lenguaje C++ se puede estar llevando la palma. Desgraciadamente no hemos podido encontrar ningún dato que se considere fiable del número de aplicaciones que están realizadas en los distintos lenguajes, sin embargo, es fácil encontrar aplicaciones realizadas, también, en VisualBasic. Cabe decir que el lenguaje Java está teniendo un gran éxito, éxito que es fácil de entender teniendo en cuenta cómo ha nacido y que permite que algo tan llevado y traído como la transportabilidad sea realmente real y no sólo algo que dicen unos papeles. Nosotros nos hemos inclinado por C++ y Java, aunque algunos proyectos los realizamos en VisualBasic y Authorware por expreso deseo de las personas que nos lo solicitan. El gran reto aquí consiste en escoger ese lenguaje adecuado, nosotros hemos indicado a cómo resolvimos nuestro reto personal, pero esto al igual que todo lo demás en informática será siempre pasajero y nos lleva al quinto y último reto del autor que es:

El mantenerse actualizado, pero sin exagerar hasta tal punto que no se termine ningún proyecto por tratar de llevarlo siempre a la última, esto , como es bien sabido en informática, es imposible, pues lo último de lo último generalmente tarda menos de un mes en dejar de serlo.

6. Conclusiones.

El desarrollo de software ha pasado por muy distintas etapas en los últimos años. Estas etapas han llevado a los autores a sentirse confundidos en la forma de trabajar y en qué recursos utilizar para que su trabajo realmente le resulte satisfactoria.

La mejor forma que puede tener un autor de sistemas educativos multimedia de desarrollar su trabajo es la que le permite hacerlo con comodidad, eficiencia y por tanto enriquecerse tanto intelectualmente como físicamente. Para conseguirlo nada mejor que seguir lo propuesto en los cinco retos de las páginas anteriores.

El cumplimiento de estos retos permitirá además que el trabajo que realice cunda en una mejora de su labor profesional. Importante es pues mantenerse al día, utilizar una metodología cercana al trabajo que se realiza, entender y utilizar siempre esa metodología, escoger el lenguaje adecuado al trabajo a realizar y no pretender estar a la última.

7. Bibliografia.

[arth91] Arthur, L.
Rapid Evolutionary Development
John Wiley (1991)

[alpi93] Alpiste, F.
Aplicaciones multimedia, presente y futuro.
Rede (1993)

[blum96] Blum, B.
Metodología de desarrollo y diseño de multimedia

[booc94] Booch, G.
Análisis y diseño orientado a objetos con aplicaciones
Addison-Wesley/Díaz de Santos(1994-2).

[chal97] Chalmenta, R.; Capuz, S.; Campos, C.; Aragones, P.
Tendencias en los proyectos de Ingeniería de Software

[coad91] Coad, P.; Yourdon, E.
Object Oriented Analysis
Yourdon Press (1991)

[coad91] Coad, P.; Yourdon, E.
Object Oriented Design
Yourdon Press (1991)

[diaz96] Díaz, P; Catenazzi, N.
De la multimedia a la hipermedia
Ra-Ma (1996)

[diaz97] Díaz Pérez, P.; Aedo Cuevas, I.; Plaza Fernández, A.
Modelos para el diseño de sistemas hipermediales
Informática y automática. Vol. 30-2/1997, pp 51-61

[fire93] Firesmith, D. G.
Object-Oriented Requirements Analysis and Logical Design
John Wiley & Sons, Inc. (1993)

[grah91] Graham, Y.
Object Oriented Methods.
Addison-Wesley (1991)

[kell92] Keller, M.
Software Specification and Design
Wiley (1992)

[pasc95] Pascual Sancho, J.
¿Cómo construir un programa multimedia?
PC World, Oct. 1995, pp 204-226

[pere79] Pérez Cota, M

Cálculo y diseño de un centro computacional de tiempo compartido con fines educativos a nivel medio y superior.
Universidad La Salle (1979)

[pere97-1] Pérez Cota, M.; Rodríguez Damián, A.; González Dacosta, J.; Vázquez Núñez, M.
MAJA, metodología para el desarrollo de aplicaciones multimedia
Universidad de Vigo (1997)

[pere97-2] Pérez Cota, M
Object Oriented Analisys, Design and Programming with C++ language
Mikkeli Polytechnic (1997-3)

[pres97] Pressman, R. S.
Ingeniería del software
Mc. Graw Hill (1997-4)

[somm88] Sommerville, I.
Ingeniería de software
Addison-Wesley Iberoamericana (1988)

[vaug96] Vaughan, T.
Todo el poder de la Multimedia Mc. Graw Hill (1996)