UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 8

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

TÍTULO: Una búsqueda de los elementos arquitectónicos del Software Educativo en la empresa Sys-COPEXTEL.

AUTOR: Leivis Salazar Salinas.

TUTOR: Ing. Dianelys Espinosa Zaldivar.

COTUTOR: M.Sc. Febe Ciudad Ricardo.

Ciudad de la Habana, Junio del 2008
"Año de 50 de la Revolución"

Ayudar al que lo necesita no solo es parte del deber, sino de la felicidad...

José Martí

A mi adorada madre y mi hermano,.

Agradecimientos

Muchas veces se nos hace difícil describir con palabras lo que el corazón siente, pero también es necesario que muchas de las personas que están a nuestro alrededor tengan una idea de lo que pensamos. Es por eso que a través de estas breves palabras quisiera hacerles llegar mi agradecimiento a cada persona que de una forma u otra ha sido parte de mi vida y me ha alentado a seguir adelante y aunque no las pueda mencionar a todas no dejan de ser importantes para mi.

A nuestro Comandante Fidel por haber creado esta hermosa escuela, en la que he pasado momentos inolvidables, donde he podido aprender muchas cosas y en la que he conocido personas de gran corazón.

A mi mamita linda a quien no tengo forma de agradecer por todo lo que me ha dado, por ser la madre más bella y buena del mundo, por ser mi sostén, mí guía, sin ti no hubiera sido capaz de convertirme en la persona que soy hoy y tratar de ser mejor. Gracias por darme valor, ayudarme a ser fuerte y responsable, por depositar tu confianza en mi. Gracias por enseñarme las cosas de la vida y ver el mundo de una manera diferente y por estar junto a mi en todos los momentos. Mamita gracias por existir porque sin ti mi vida no tendría sentido.

A mi hermano que con su amor y cariño me motivo a llegar a donde estoy hoy. Por ser tan especial te agradesco que siempre pueda contar contigo. A mi sobrinita por su ternura y cariño.

Ami abuela por apoyarme en todo y ser muy paciente conmigo. Por entenderme siempre y por amarme tanto.

A Yannelys, Lizzet, Surelys, Suri, Yiliana, Albertini, Frank, Yuliet, Eradys, Antonio, Santiago por estar conmigo simpre que los necesite en las buenas y las malas. Por darme mucho apoyo y por quererme tanto.

A mi titi que ha sabido llenarme de amor y cariño en tan poco tiempo. Por ser tan especial, cariñoso, dedicado, paciente y siempre estár dispuesto para lo que yo necesite.

A todos mis compañeros del aula por los buenos momentos que pasamos juntos.

En fin a todos los que de una forma u otra me ayudaron y me dieron todo el apoyo que se necesita para poder lograr grandes cosas gracias y mil gracias .

Declaración de Autoría

Declaro ser autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de Junio del año 2008.

Leivis Salazar Salinas	Ing. Dianelys Espinosa Zaldivar
_____	_____
Firma del Autor	Firma del Tutor

Resumen

Las Tecnologías de la Información y las Comunicaciones (TIC), conllevan cambios que alcanzan todos los ámbitos de la actividad humana. Sus efectos se manifiestan en varios terrenos y fundamentalmete en el educativo donde es cada día mayor. Debido a esto, en un período relativamente joven Sys_ COPEXTEL ha organizado el proceso de producción del software e introducido los mismos en todas las educaciones. Es por ello que el trabajo que se expone a continuación esta centrado en una propuesta de solución de los elementos arquitectónicos para software educativo producido en esta institución. Analizando la fuerte línea de desarrollo de productos Software Educativo en Sys_COPEXTEL se define el objetivo concreto de este trabajo, que constituye el planteamiento de una propuesta de los elementos arquitectónicos a utilizar en el proceso de desarrollo de estos productos, garantizando con esta propuesta un avance hacia la mejora de la producción de software en Sys_COPEXTEL, aumentando las expectativas de confiabilidad de los clientes, así como la satisfacción de las necesidades de los mismos y unido a esto, se incrementaría la ventaja competitiva de la organización frente a sus competidores.

Palabras Claves
Software educativo, elementos arquitectónicos, arquitectura de software.

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Introducción

Cuba sostiene que la Tecnología no es neutral, responde siempre a los intereses de quienes la poseen y la aplican. Esta es una las explicaciones de por qué la extensión de las TIC por el mundo, con un enorme potencial de beneficio, paradójicamente ha contribuido con la brecha digital a acentuar la brecha socioeconómica entre ricos y pobres, entre poseedores y desposeídos, entre explotadores y explotados. Cuba ha defendido siempre el concepto de que el uso masivo de las TIC no es un fin sino una herramienta poderosa para lograr el desarrollo. A raíz de estos cambios se comienza a utilizar los sistemas con tecnología multimedia, debido a su fácil manejo, portabilidad e interactividad.

Tras el triunfo revolucionario, Cuba se propuso un camino de desarrollo que pudiera solucionar por igual las necesidades materiales básicas y las espirituales de su población, sobre la base de una distribución más justa y equitativa de la riqueza. De esa forma, se logró satisfacer, con un acceso universal, las necesidades primarias de salud, educación, deporte, empleo, desarrollo cultural, libertad y participación política, protección y asistencia social, a la vez que fueron emprendidas varias líneas de desarrollo científico-técnico que en algunas ramas la han situado en un lugar destacado a nivel mundial.

Cuba sostiene la idea de que a la sociedad le es necesario universalizar el conocimiento como una de las formas de alcanzar una mejor calidad de vida para todos los ciudadanos, sin distinción de edad ni condición social. La fórmula “educación para todos, durante toda la vida”, se presenta como el núcleo de un amplio movimiento educacional que abarca todo el país y a todos los ciudadanos.

Para lograr este macro objetivo de universalizar el conocimiento, se trabajó en 2 grandes vertientes, el Perfeccionamiento de la Enseñanza General y la Universalización de la Universidad. El 100% de los centros de la enseñanza primaria, secundaria, tecnológica y universitaria del país usan las TIC como apoyo a los programas de clases. Los objetivos fundamentales que persigue el uso de esta tecnología son: elevar la calidad de la educación cubana, garantizar la necesaria preparación en las TIC de los recursos humanos, instrumentar un proceso de educación continua y ampliar la cultura general de la población sobre estas tecnologías.

En Cuba se acumulan más de 20 años de experiencia en la elaboración de software educativo. En estos resultados han tenido una importante participación los Ministerios de Educación y Educación Superior por medio de los centros de estudio del software y las Universidades respectivamente, así como la organización de los Joven Club de conjunto con el sector empresarial.

A lo largo del tiempo mencionado aparecieron en la industria internacional de software metodologías como la Relationship Management Methodology (RMM) para la guía de procesos de desarrollo de aplicaciones con tecnología multimedia; la técnica más utilizada en el desarrollo de aplicaciones de corte educativo, que anexaban una notación para el modelado de este tipo de software. Cumplió su objetivo durante un buen tiempo, pero tecnológicamente no logra responder a todas nuestras necesidades actuales en la representación computacional de las soluciones informáticas educativas. De manera similar, en Cuba se trabajó en nuestras necesidades surgiendo la metodología MultiMet, que igualmente traía asociada un lenguaje de representación de este tipo de software.

Al mismo tiempo, como ya fue mencionado, evolucionaron las técnicas y lenguajes de programación, y con estas mejoraron las formas de programar el software educativo, llegando en la actualidad a utilizar en mayor medida técnicas orientadas a eventos y en sus mejores exponentes orientados a objetos completamente.

La presentación en el año 1998 por sus autores: I. Jacobson, G. Booch y J. Rumbaugh de la primera versión del Lenguaje Unificado de Modelado (UML) para la modelación de software de propósito general orientado a objetos, fue de un gran impacto en la industria internacional y tuvo una excelente aceptación, al punto de convertirse en un estándar mundial en este sentido. “Desafortunadamente, UML no soporta todos los aspectos de las aplicaciones multimedia de una forma adecuada e intuitiva. Especialmente, las características del lenguaje para modelar los aspectos de la interfaz de usuario, no se aplican explícitamente en los entornos multimedia. Otros conceptos de UML no son lo formalmente aplicables a la multimedia y de ser utilizados tal y como han sido planteados complicarían la modelación de este tipo de aplicaciones.

Si señalamos además que de los diferentes tipos de software educativo existentes según la clasificación ofrecida por (Marqués, 1996): tutoriales, bases de datos, simuladores, constructores y programas herramienta; el software educativo cubano es un producto que fusiona elementos

representativos de cada uno de los tipos específicos mencionados anteriormente; aumentando considerablemente la complejidad del tipo de software y su modelación, hecho en Cuba.

Teniendo en cuenta que la empresa que ocupa el centro de la investigación es Sys-COPEXTEL y las principales funciones de sus especialistas son: convertir en modelos ingenieriles informáticos, con capacidad de diseño e implementación, las necesidades y requerimientos de los software educativos actuales, se hace necesario la utilización de un lenguaje notacional que logre representar en modelos la estructura lógica, el comportamiento y las funciones del futuro software a desarrollar.

La Empresa Sys-COPEXTEL aunque lleva varios años de creada y su grupo de trabajadores está altamente preparado para la producción de software, trabaja con la metodología RUP y con el lenguaje de modelado UML.

Teniendo en cuenta que la metodología utilizada por Sys-COPEXTEL con éste lenguaje de modelado no permite representar la totalidad de los artefactos que se necesitan para la elaboración de su software de una forma adecuada e intuitiva y que no cuenta con una arquitectura definida para realizar los mismos se identificó como **problema científico** de esta investigación: Insuficiente conocimiento de los elementos estructurales, lógicos comunes y significativos del software educativo en la empresa Sys – COPEXTEL.

Como **Idea a Defender** de la presente investigación se plantea la siguiente: La aplicación de los elementos estructurales y funcionales (arquitectónicos) significativos en la producción del software educativo en la empresa Sys – COPEXTEL, permitirá un mejor diseño funcional y pedagógico, así como una disminución del tiempo de desarrollo de este tipo de aplicación en la empresa mencionada.

Esto genera como **objeto de estudio** de esta investigación: Procesos de concepción y diseño pedagógico del software educativo.

Por lo que el **campo de acción** de la investigacion es: Proceso de concepción y diseño pedagógico del software educativo en la empresa Sys – COPEXTEL.

Para dar solución al problema científico planteado se propuso como **objetivo general**: Proponer los elementos estructurales y lógicos (arquitectónicos) a utilizar en la producción de software educativo en la empresa Sys – COPEXTEL.

Seguidamente se plantean los siguientes **objetivos específicos** para lograr un mejor resultado del objetivo general descrito anteriormente:

1. Definir los componentes conceptuales comunes, significativos y con capacidad de generalización de los elementos estructurales y lógicos del software educativo producido en la empresa Sys – COPEXTEL.
2. Definir los elementos arquitectónicos necesarios en la producción del software educativo en la empresa Sys – COPEXTEL.

Las tareas por las que se regirá la investigación para dar cumplimiento tanto a los objetivos generales como a los específicos son:

1. Estudiar los elementos estructurales y funcionales significativos en la producción de software educativo en el mundo actualmente.
2. Estudiar el estado actual de la producción de software educativo en Cuba.
3. Estudiar la documentación generada a partir del diseño y concepción pedagógica del software educativo en la empresa Sys – COPEXTEL.
4. Identificar los elementos funcionales comunes y significativos como resultado de los diferentes procesos de diseño y concepción pedagógica utilizados en el mundo.
5. Identificar los elementos estructurales comunes y significativos como resultado de los diferentes procesos de diseño y concepción pedagógica utilizados en la empresa Sys – COPEXTEL.
6. Identificar y evaluar los elementos pedagógicos significativos y comunes en los softwares educativos producidos en la empresa Sys – COPEXTEL.
7. Validar en un proyecto productivo del Sys – COPEXTEL la aplicación de los elementos propuestos.

En el desarrollo de la investigación científica se han utilizado un conjunto de métodos científicos para la obtención, procesamiento y llegada a conclusiones. Dentro de estos podemos mencionar los siguientes:

• Métodos Teóricos:

1. Hipotético – deductivo: para la determinación de la idea a defender de esta investigación y proponer nuevas líneas de trabajo a partir de los resultados parciales.
2. Sistémico: para lograr que los diferentes elementos que forman parte del entorno de la notación descrita funcionen de manera armónica e igualmente garantizar el acoplamiento con otros lenguajes o notaciones ya existentes.
3. Histórico – lógico y Dialéctico: para el estudio crítico de los trabajos anteriores y para utilizar estos como punto de referencia y comparación de los resultados alcanzados.
4. Analítico – sintético: al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.

• Métodos Empíricos:
1. Observación: para obtener el registro del avance del proyecto productivo que sirvió de muestra para el caso de estudio desarrollado, en la clasificación y consignación de los acontecimientos pertinentes para la investigación.
2. Entrevista: utilizado para obtener información verbal y real de distintos miembros internos y externos a la actividad de modelación en el proyecto, así como a los especialistas pedagógicos inmersos en la investigación.
3. Encuesta: para conformar las estadísticas de la situación actual con las notaciones existentes, así como las de impacto de la solución propuesta en la investigación.

El presente Trabajo de Diploma consta de una Introducción, 3 capítulos con la intención de realizar una división por los contenidos que serán tratados, las conclusiones generales, recomendaciones, referencias bibliográficas utilizadas durante el desarrollo del trabajo, un glosario de términos y siglas y por último los anexos que complementan el cuerpo del trabajo y que son necesarios para su entendimiento.

Capítulo 1: Fundamentación Teórica. En este capítulo se hace un análisis de la actualidad internacional y nacional sobre el tema de los diferentes tipos de arquitecturas del software profundizando en los términos referentes al mismo, además se abordan las principales definiciones que se tiene en cuenta durante todo el trabajo.

Capítulo 2: Descripción de la Propuesta de Solución. En este capítulo se presentan los elementos que servirán de fundamento teórico a la investigación y se presenta el patrón de arquitectura propuesto, y se describen cada uno de sus elementos .

Capítulo 3: Validación de la Propuesta de Solución. En este capítulo se determina la probabilidad de éxito que tiene la propuesta, para esto se realizan un conjunto de cálculos los cuales se detallan con el objetivo de lograr un mayor entendimiento.

Capítulo 1: “Fundamentación Teórica”

1.1 Introducción

En este capítulo se abordarán los elementos que servirán de fundamento teórico a la investigación presentada. Se hace referencia a conceptos teóricos que se relacionan con la situación problemática que sirve como fundamentación científica de lo investigado.

1.2 Arquitectura de software

En los inicios de la informática, la programación se consideraba un arte, debido a la dificultad que entrañaba para la mayoría de los mortales, pero con el tiempo se han ido desarrollando metodologías para conseguir esos propósitos. Y a todas estas técnicas se les ha dado en llamar Arquitectura de Software.

Existen diferentes definiciones de lo que es la Arquitectura de Software y no parece que ninguna de ellas ha sido totalmente aceptada.

• Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.
• La Arquitectura de Software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.
• Una arquitectura de software se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas

arquitecturas. Por ejemplo, no es viable emplear una arquitectura de software de tres capas para implementar sistemas en tiempo real.

• La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación ente ellos. Toda arquitectura debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea.

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad. (Kruchten, Philippe)

1.2.1 Principales corrientes arquitectónicas

Arquitectura estructural, basada en un modelo estático de estilos, ADLs y vistas. Constituye la corriente fundacional y clásica de la disciplina. Los representantes de esta corriente son todos académicos, mayormente de la Universidad Carnegie Mellon en Pittsburgh: Mary Shaw, Paul Clements, David Garlan, Robert Allen, Gregory Abowd, John Ockerbloom. En toda la corriente, el diseño arquitectónico no sólo es el de más alto nivel de abstracción, sino que además no tiene por qué coincidir con la configuración explícita de las aplicaciones; rara vez se encontrarán referencias a lenguajes de programación o piezas de código.

Arquitectura como una Etapa de ingeniería y diseño orientada a objetos. Esta corriente es una alternativa de la descrita anteriormente. Es el modelo de James Rumbaugh, Ivar Jacobson, Grady Booch, Craig Larman y otros, ligado estrechamente al mundo de UML y Rational. En esta postura, la arquitectura se restringe a las fases iniciales y preliminares del proceso y concierne a los niveles más elevados de abstracción. Importa más la abundancia y el detalle de diagramas y técnicas disponibles que la simplicidad de la visión de conjunto. La definición de arquitectura que se promueve en esta corriente tiene que ver con aspectos formales a la hora del desarrollo. Las definiciones revelan que la AS, en esta perspectiva, concierne a decisiones sobre organización, selección de elementos estructurales, comportamiento, composición y estilo arquitectónico susceptibles de ser descritas a través de las cinco vistas clásicas del modelo 4+1 de Kruchten, que el mismo es el modelo más

aceptado a la hora de establecer las vistas necesarias para describir una arquitectura software (Figura 1) .

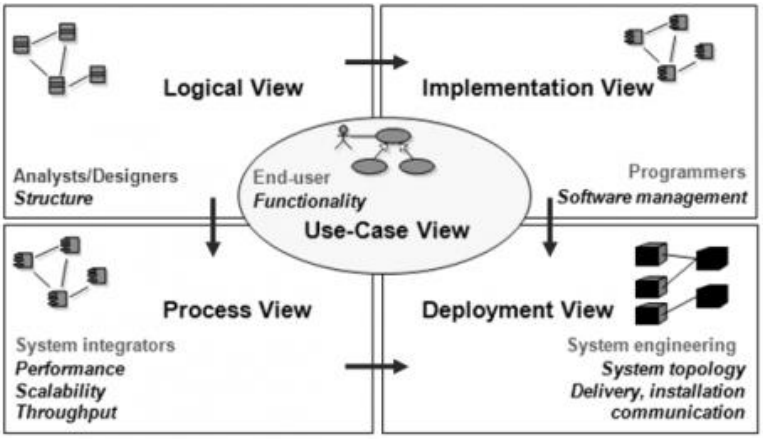

Figura 1. Vistas Arquitectónicas

Arquitectura basada en patrones, esta corriente se basa principalmente en la redefinición de los estilos como patrones POSA (Patrones de arquitectura orientada a servicios), el diseño consiste en identificar y articular patrones preexistentes,

Arquitectura procesual y metodologías. Esta surge desde comienzos del siglo XXI, con centro en el SEI y con participación de algunos arquitectos de Carnegie Mellon de la primera generación y muchos nombres nuevos de la segunda: Rick Kazman, Len Bass, Paul Clements, Félix Bachmann, Fabio Peruzzi, Jeromy Carrière, Mario Barbacci y Charles Weinstock. Intenta establecer modelos de ciclo de vida y técnicas de diseño, análisis, selección de alternativas, validación, comparación, estimación de calidad y justificación económica específicas para la arquitectura de software.

1.2.2 Modelos o Vistas

Toda arquitectura de software debe describir diversos aspectos del software. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. Es importante destacar que cada uno de ellos constituye una descripción parcial de una misma

arquitectura y es deseable que exista cierto solapamiento entre ellos. Esto es así porque todas las vistas deben ser coherentes entre sí, evidente dado que describen la misma cosa.

Cada paradigma de desarrollo exige diferente número y tipo de vistas o modelos para describir una arquitectura. No obstante, existen al menos tres vistas absolutamente fundamentales en cualquier arquitectura:

- La visión **estática**: describe qué componentes tiene la arquitectura.
- La visión **funcional**: describe qué hace cada componente.
- La visión **dinámica**: describe cómo se comportan los componentes a lo largo del tiempo y como interactúan entre sí.

Las vistas o modelos de una arquitectura pueden expresarse mediante uno o varios lenguajes. El más obvio es el lenguaje natural, pero existen otros lenguajes tales como los diagramas de estado, los diagramas de flujo de datos, etc. Estos lenguajes son apropiados únicamente para un modelo o vista. Afortunadamente existe cierto consenso en adoptar UML (Unified Modeling Language, lenguaje unificado de modelado) como lenguaje único para todos los modelos o vistas. Sin embargo, un lenguaje generalista corre el peligro de no ser capaz de describir determinadas restricciones de un sistema de información (o expresarlas de manera incomprensible).

1.2.3 Arquitecturas más comunes

Generalmente, no es necesario inventar una nueva arquitectura de software para cada sistema de información. Lo habitual es adoptar una arquitectura conocida en función de sus ventajas e inconvenientes para cada caso en concreto. Así, las arquitecturas más universales son:

- Monolítica. Donde el software se estructura en grupos funcionales muy acoplados.
- Cliente-servidor. Donde el software reparte su carga de cómputo en dos partes independientes pero sin reparto claro de funciones.
- Arquitectura de tres niveles. Especialización de la arquitectura cliente-servidor donde la carga se divide en tres partes (o capas) con un reparto claro de funciones: una capa para la presentación (interfaz de usuario), otra para el cálculo (donde se encuentra modelado el negocio) y otra para el almacenamiento (persistencia).

- Una capa solamente tiene relación con la siguiente.

Otras arquitecturas menos conocidas son:

- En pipeline.
- Entre pares.
- En pizarra.
- Orientada a servicios.
- Máquinas virtuales.

1.3 La informática y el software, al servicio de la educación

En las últimas tres décadas se ha desarrollado un vertiginoso desarrollo de la incorporación de la informática al entorno educativo, dando surgimiento al área de la Informática Educativa y con ella a conceptos como Software Educativo, que se hace muy común en nuestros días. El propio desarrollo de este tipo de software ha traído como consecuencia variaciones importantes en los flujos de los procesos productivos y en los artefactos de modelación para generar su documentación en el uso de las tecnologías multimedia e hipermedia.

Según la enciclopedia Microsoft Encarta'97 la multimedia se puede clasificar como una forma de presentar información, en su combinación de texto, sonido, imágenes, animaciones y video. Ejemplos de aplicaciones multimedia informáticas son: juegos interactivos, programas de aprendizaje, materiales de referencia, por ejemplo enciclopedias. En los últimos años, varios autores han intentado conceptualizar la tecnología multimedia. Pero una concepción multifocal de la multimedia es la que plantea Rodríguez Lamas: "Es una nueva plataforma donde se integran componentes para hacer ciertas tareas que proporcionan a los usuarios nuevas oportunidades de trabajo y acceso a nuevas tecnologías. Es un nuevo medio donde la computadora junto con los medios tradicionales dan una nueva forma de expresión. Es una nueva experiencia donde la interacción con los medios es radicalmente diferente y donde tenemos que aprender cómo usarlos. Es una nueva industria donde, con una nueva plataforma, un nuevo medio y una nueva experiencia, nos llevan a tener nuevas oportunidades de negocios." (Rodríguez, 2000)

La tecnología multimedia aparece en el año 1984, cuando la Apple Computer lanza la primera variante de Macintosh, la cual tenía amplias capacidades de reproducción de sonidos; apoyándose en su sistema operativo, propicio para el diseño gráfico y la edición. En 1987, con los videojuegos, los avances en las Tecnologías de la Información y las Comunicaciones (TIC) y el desarrollo por la Philips del Disco Compacto (CD), aumentan las posibilidades para el desarrollo de la tecnología multimedia.

En 1992, se presenta en la feria "Consumer Electronic Show" (CES) de Las Vegas, el CD multiusos, que luego dió paso a la televisión interactiva. En 1993, el nuevo término de multimedia, obliga a una revisión de los conceptos que permiten el funcionamiento de la nueva tecnología, buscando el desarrollo de estándares para lograr la uniformidad en el desarrollo. Hoy se ha desarrollado una nueva técnica que surgió a partir de la multimedia: la hipermedia, que tiene su base en el hipertexto, permitiendo al usuario un recorrido o navegación por distintos archivos o partes del programa de acuerdo a sus intereses personales. La multimedia tiene varias aplicaciones entre las cuales se pueden mencionar las siguientes: (Corrales Díaz, 1994)

• En la diversión y el entretenimiento: donde se sitúan los juegos de video y las aplicaciones de pasatiempos de tipo cultural.

• Multimedia en los negocios: sus principales exponentes están en los kioscos de información, las presentaciones, intercambio y circulación de información.

• En publicidad y marketing: sus ejemplares son; la presentación multimedia de negocios, de productos y servicios, la oferta y difusión de los productos y servicios a través de los kioscos de información.

• En la difusión del saber y conocimiento: La característica de la interactividad de multimedia, que permite navegar por el programa y buscar la información sin tener que recorrerlo todo, logra que la tecnología se aplique en los nuevos medios de modos diferentes y se use de forma alternativa.

Y por último entre los muchos beneficios que ofrecen la tecnología multimedia se puede mencionar: el impacto al incorporar imágenes, efectos de sonido, video y animación en tercera dimensión para crear presentaciones vivas y de extraordinaria calidad. La flexibilidad, ya que el material digital puede ser fácil y rápidamente actualizado y presentado a través de innumerables medios. El control por parte del emisor, al seleccionar la cantidad y tipo de información que desea entregar así como la forma de entregarla al igual que el control por parte del receptor, al elegir la información que quiere recibir y en el momento en que desea recibirla. El ahorro de recursos en materiales impresos difíciles de actualizar y presentándola en innumerables ocasiones sin ninguna restricción.

1.3.1 La tecnología multimedia

A continuación se dará una serie de datos y características que permitirán conocer un poco más a la tecnología hipermedia.

1.3.2 La tecnología hipermedia

Vannevar Bush, en la década de los 40, y Theodor Holme Nelson, en los 60, se consideran los artífices de la estructuración no lineal y de la interconexión de la información, asuntos que constituyen conceptos claves para el desarrollo de la interactividad informática aplicada a la comunicación. El hipertexto no es más que un conjunto de bloques de texto interconectados por nexos, que forman diferentes itinerarios para el usuario, donde estos nexos "electrónicos" unen fragmentos te texto internos o externos a la obra (soportes cerrados – off line – o abiertos – on line – respectivamente), creando un texto que el lector experimenta como no lineal o, mejor dicho, como ultilineal o multisecuencial.

La hipermedia surge como resultado de la fusión de dos tecnologías, el hipertexto y la multimedia. La tecnología multimedia es la que permite integrar diferentes medios (sonido, imágenes, secuencias...) en una misma presentación. La hipermedia, por tanto, es la tecnología que permite estructurar la información de una manera no-secuencial, a través de nodos interconectados por enlaces sobre los diferentes formatos de información.

Estas hipermedias y multimedias pretenden resolver el problema del procesamiento lineal de la información por el receptor, como ocurre en el libro de texto. Por el contrario, la información se puede construir desde diferentes trayectorias y alternativas, y con diferentes tipos de códigos. Estas trayectorias pueden limitarse por el autor del programa, para evitar problemas de desorientación en el usuario (Pastor, 1997), pudiendo soportar la autoría de documentos complejos y el procesamiento de ideas, especialmente en entornos de trabajo colaborativos. Para la aplicación de esta facilidad los alumnos necesitan la capacidad de anotar nodos de información y ser capaces de colaborar con otros estudiantes y con el profesor sobre unidades específicas de información. Aún más importante que esto es que deben ser capaces de construir su propio sistema de conocimiento a través de la investigación, abstracción, readaptación y adición de conocimientos a una base de datos existentes.

1.3.3 La Informática y las aplicaciones educativas

A continuación se hace un análisis a fondo de una aplicación de las tecnologías multimedia e hipermedia: el Software Educativo. “Todas las formas de software educativo han sido absorbidas por esta tecnología, lo cual no es pura casualidad, sino el resultado de un proceso histórico que ha pretendido combinar los diferentes métodos para transmitir la información, en esperanza de una mayor calidad del propio proceso de adquisición de conocimientos.” (Lee, y otros, 2000)

“Las Tecnologías de la Información y la Comunicación (TIC) ofrecen grandes posibilidades al mundo de la educación. Pueden facilitar el aprendizaje de conceptos y materias, ayudar a resolver problemas y contribuir a desarrollar las habilidades cognitivas. Las áreas de aplicación de todas estas técnicas, englobadas en lo que normalmente se denomina informática educativa, son tanto la enseñanza reglada, comúnmente denominada curricular, como la formación en todos los ámbitos posibles. De esta manera, se nos presenta la posibilidad de aprovechar la tecnología para crear situaciones de aprendizaje y enseñanza novedosas. “(Diáz-Antón, y otros, 2002)

Una de las clasificaciones más conocida fue dada por los norteamericanos Stephen M. Alessi y Stanley Trollip, cuando plantearon que el uso de las computadoras en la educación podía dividirse en: (Pérez Fernández, 1999)

- Uso administrativo.
- Enseñanza sobre computadoras.
- Enseñanza con computadoras, dentro de la cual se enmarca mayoritariamente el área de la informática educativa.

Según Raúl Rodríguez Lamas, la Informática Educativa se puede definir como la parte de la ciencia de la Informática encargada de dirigir, en el sentido más amplio, todo el proceso de selección, elaboración, diseño y explotación de los recursos informáticos dirigido a la gestión docente, entendiéndose por éste la enseñanza asistida por computadora y la administración docente. (Rodríguez, 2000)

La verdadera revolución se ha producido a raíz de la generalización de los CD-ROMs como soporte de información para las aplicaciones multimedia y al mismo tiempo una auténtica revolución en el software, sobre todo en la interfaz de usuario. Vinculando la enseñanza por computadora con las nuevas tecnologías multimedia, surge lo que conocemos como los software multimedia educativos; herramientas poderosas dentro del contexto de la Informática Educativa.

Según Vicenta Pérez Fernández un software multimedia educativo es "una aplicación informática, soportada sobre una bien definida estrategia pedagógica, y que apoya directamente el proceso de enseñanza-aprendizaje constituyendo un efectivo instrumento para el desarrollo educacional del hombre del próximo siglo." (Pérez Fernández, 1999)

Los software multimedia educativos, permiten agrupar una serie de factores presentes en otros medios, pero a la vez agregar otros hasta ahora inalcanzables: (Pérez Fernández, 1999) (Fernández, 2000)

• Permite la interactividad con los estudiantes, retroalimentándolos y evaluando lo aprendido.
• Facilita las representaciones animadas.
• Incide en el desarrollo de las habilidades a través de la ejercitación. Permite simular procesos complejos.
• Reduce el tiempo que se dispone para impartir gran cantidad de conocimientos facilitando un trabajo diferenciado, introduciendo al estudiante en el trabajo con los medios computarizados.
• Facilita el trabajo independiente y a la vez un tratamiento individual de las diferencias.
• Permite al usuario (estudiante) introducirse en las técnicas más avanzadas.
• Posibilidades de estudiar procesos que no es posible observar directamente.
• Autocontrol del ritmo de aprendizaje.

Concluyendo, los buenos software multimedia formativos dentro del marco de la Informática Educativa, son eficaces y facilitan el logro de sus objetivos, debido al supuesto buen uso por parte de los estudiantes y profesores, a una serie de características que atienden a diversos aspectos funcionales, técnicos y pedagógicos. A continuación se mencionan algunos: (Autores, 1998)

• Facilidad de uso e instalación.
• Versatilidad (adaptación a diversos contextos).
• Calidad del entorno audiovisual.
• La calidad en los contenidos (bases de datos).

• Navegación e interacción.
• Originalidad y uso de tecnología avanzada.
• Capacidad de motivación.
• La documentación.

1.4 Conceptos asociados al dominio del problema

Para un mejor entendimiento del objetivo de la investigación existen una serie de conceptos que son fundamentales para ubicarse en el entorno de lo que se quiere explicar.

1.4.1 Software educativo

Sánchez J. (1999), en su libro "Construyendo y Aprendiendo con el Computador", define al software educativo como un programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar, aprender y administrar. Un concepto más restringido de software educativo lo define como aquel material de aprendizaje especialmente diseñado para ser utilizado con un computador en los procesos de enseñar y aprender.(Álvarez and Rodríguez 2006) La definición anterior es reflejada en al artículo: “Software Educativo. Su influencia en la escuela cubana”, cuyas autoras expresan posteriormente: “Finalmente, los Software Educativos se pueden considerar como el conjunto de recursos informáticos diseñados con la intención de ser utilizados en el contexto del proceso de enseñanza – aprendizaje. Se caracterizan por ser altamente interactivos, a partir del empleo de recursos multimedia, como videos, sonidos, fotografías, diccionarios especializados, explicaciones de experimentados profesores, ejercicios y juegos instructivos que apoyan las funciones de evaluación y diagnóstico.” (Álvarez and Rodríguez 2006)

El software educativo está formado por los programas educativos y didácticos creados con la finalidad específica de ser utilizados para facilitar los procesos de enseñanza – aprendizaje, como se había reflejado en las definiciones anteriores, es decir, tiene la finalidad de lograr que el estudiante adquiera conocimientos, habilidades, procedimientos, y esto es posible gracias a la interactividad que surge entre el estudiante y el computador, a través del cual se puede individualizar el trabajo de ese estudiante, pues este tipo de software permite adaptar el ritmo de trabajo a cada uno, así como las actividades que tiene implícitas según la actuación de los mismos. Es importante destacar que existen programas que son utilizados en los centros educativos con funciones didácticas o instrumentales, que aunque pueden desarrollar dichas funciones, no han sido elaborados específicamente con esa

finalidad, por tanto se considera que está excluido de la categoría de software educativo. Ejemplo de ello tenemos a los procesadores de textos, gestores de bases de datos, hojas de cálculo, editores gráficos, entre otros.

1.4.2 Proceso de enseñanza – aprendizaje

"El **proceso de enseñanza – aprendizaje** constituye la vía mediatizadora esencial para la apropiación de conocimientos, habilidades, hábitos, normas de relación, de comportamiento y valores, legados por la humanidad, que se expresan en el contenido de enseñanza, en estrecho vínculo con el resto de las actividades docentes y extra docentes que realizan los estudiantes". (J. Zilberstein, 1999) "Enseñanza y aprendizaje forman parte de un único proceso que tiene como fin la formación del estudiante". (Mancebo, 1999) "El proceso de enseñanza – aprendizaje es un proceso esencialmente interactivo y comunicativo, de intercambio de información, compartiendo experiencias, conocimientos y vivencias, que logran una influencia mutua en las relaciones interpersonales". (Fernández, 2000) El objetivo final de este proceso es la formación del estudiante y esto se logrará a través de como el profesor muestra a sus alumnos los contenidos, hábitos y habilidades, utilizando medios y con la intención de lograr objetivos fijados desde un inicio en un determinado contexto.

1.5 Lenguajes de descripción arquitectónica

Los lenguajes de descripción de arquitecturas (ADLs) ocupan una parte importante del trabajo arquitectónico desde la fundación de la disciplina. Se trata de un conjunto de propuestas de variado nivel de rigurosidad, casi todas ellas de extracción académica, que fueron surgiendo desde comienzos de la década de 1990 hasta la actualidad, más o menos en contemporaneidad con el proyecto de unificación de los lenguajes de modelado bajo la forma de UML.

Los ADLs permiten modelar una arquitectura mucho antes que se lleve a cabo la programación de las aplicaciones que la componen, analizar su adecuación, determinar sus puntos críticos y eventualmente simular su comportamiento.

1.5.1 Acme – Armani

Acme se define como una herramienta capaz de soportar el mapeo de especificaciones arquitectónicas entre diferentes ADLs, o en otras palabras, como un lenguaje de intercambio de arquitectura. No es entonces un ADL en sentido estricto, aunque la literatura de referencia acostumbra tratarlo como tal. De hecho, posee numerosas prestaciones que también son propias de los ADLs. En su sitio oficial se reconoce que como ADL no es necesariamente apto para cualquier clase de sistemas, al mismo tiempo que se destaca su capacidad de describir con facilidad sistemas “relativamente simples”. (Figura 2)

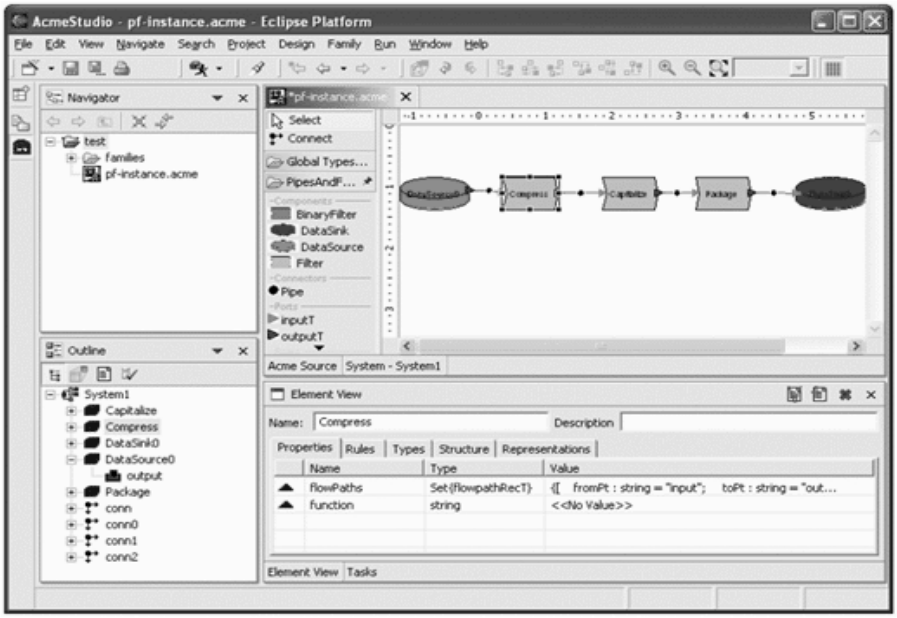

Figura 2. Ambiente de edición de AcmeStudio con diagrama de tubería y filtros

Objetivo principal

La motivación fundamental de Acme es el intercambio entre arquitecturas e integración de ADLs. Acme soporta la definición de cuatro tipos de arquitectura: la estructura (organización de un sistema en sus partes constituyentes); las propiedades de interés (información que permite razonar sobre el comportamiento local o global, tanto funcional como no funcional); las restricciones (lineamientos sobre la posibilidad del cambio en el tiempo); los tipos y estilos. La estructura se define utilizando siete tipos de entidades: componentes, conectores, sistemas, puertos, roles, representaciones y rep-mapas (mapas de representación).

Componentes

Representan elementos computacionales y almacenamientos de un sistema.

Interfaces

Todos los ADLs conocidos soportan la especificación de interfaces para sus componentes. En Acme cada componente puede tener múltiples interfaces.

Conectores

En catalogado como un lenguaje de configuración explícito, pues, modela sus conectores como entidades de primera clase y representan interacciones entre componentes.

Semántica

Muchos lenguajes de tipo ADL no modelan la semántica de los componentes más allá de sus interfaces. En este sentido, Acme sólo soporta cierta clase de información semántica en listas de propiedades. Estas propiedades no se interpretan, y sólo existen a efectos de documentación.

Estilos

Acme posee manejo intensivo de familias o estilos. Esta capacidad está construida naturalmente como una jerarquía de propiedades correspondientes a tipos. Acme considera, en efecto, tres clases de tipos: tipos de propiedades, tipos estructurales y estilos. Así como los tipos estructurales representan conjuntos de elementos estructurales, una familia o estilo representa un conjunto de sistemas. Una familia Acme se define especificando tres elementos de juicio: un conjunto de tipos de propiedades y tipos estructurales, un conjunto de restricciones y una estructura por defecto, que prescribe el conjunto mínimo de instancias que debe aparecer en cualquier sistema de la familia. La biblioteca AcmeLib define un conjunto de clases para manipular representaciones arquitectónicas Acme en cualquier aplicación. Su código se encuentra disponible tanto en C++ como en Java, y puede ser invocada por lo tanto desde cualquier lenguaje la plataforma clásica de Microsoft o desde el framework de .NET. Es posible entonces implementar funcionalidad conforme a Acme en forma directa en cualquier programa, o llegado el caso (mediante wrapping) exponer Acme como web service.

Interfaz gráfica

La versión actual de Acme soporta una variedad de front-ends de carácter gráfico, de los cuales he experimentado con tres. El ambiente primario, llamado AcmeStudio, es un entorno gráfico basado en Windows, susceptible de ser configurado para soportar visualizaciones específicas de estilos e invocación de herramientas auxiliares. Un segundo entorno, llamado Armani, utiliza Microsoft Visio como front-end gráfico y un back-end Java, que con alguna alquimia puede ser Microsoft Visual J++ o incluso Visual J# de .NET. Un tercer ambiente, más experimental, utiliza el editor de PowerPoint para manipulación gráfica.

Generación de código

En los últimos años se ha estimado cada vez más deseable que un ADL pueda generar un sistema ejecutable, aunque más no sea de carácter prototípico. De tener que hacerlo manualmente, se podrían suscitar problemas de consistencia y trazabilidad entre una arquitectura y su implementación. Acme, se concibe como una notación de modelado y no proporciona soporte directo de generación de código.

Disponibilidad de plataforma

Ya me he referido a AcmeStudio (2.1), un front-end gráfico programado en Visual C++ y Java que corre en plataforma Windows y que proporciona un ambiente completo para diseñar modelos de arquitectura.

1.5.2 Aesop

La definición también oficial de Aesop es “una herramienta para construir ambientes de diseño de software basada en principios de arquitectura”. El ambiente de desarrollo de Aesop System se basa en el estilo de tubería y filtros propio de UNIX. (Figura 3)

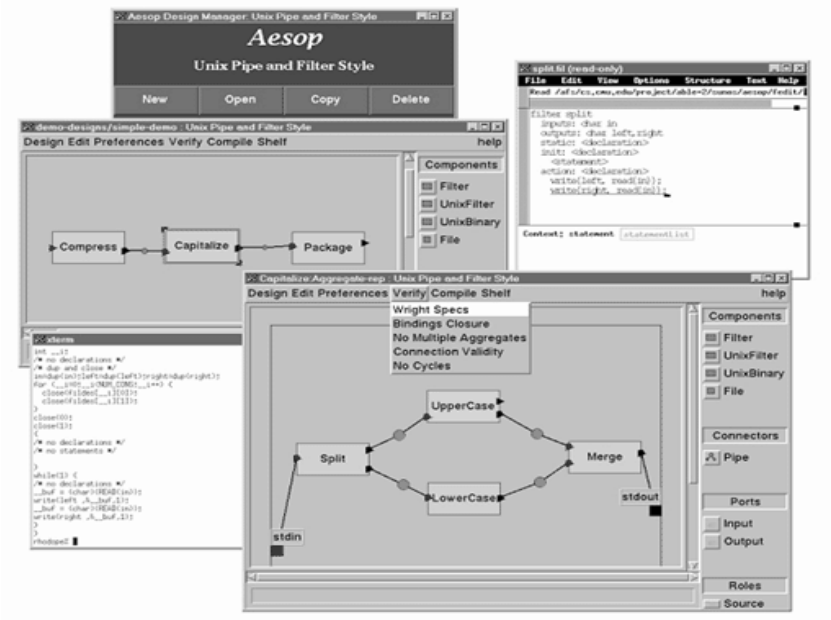

Figura 3. Ambiente gráfico de Aesop con diagrama de tubería y filtro

Estilos

En Aesop, conforme a su naturaleza orientada a objetos, el vocabulario relativo a estilos arquitectónicos se describe mediante la definición de sub-tipos de los tipos arquitectónicos básicos: Componente, Conector, Puerto, Rol, Configuración y Binding.

Interfaces

En Aesop los puntos de interfaz se llaman puertos (ports).

Modelo semántico

Aesop presupone que la semántica de una arquitectura puede ser arbitrariamente distinta para cada estilo. Por lo tanto, no incluye ningún soporte nativo para la descripción de la semántica de un estilo o configuración, sino que apenas presenta unos cuadros vacantes para colocar esa información como comentario.

Soporte de lenguajes

Aesop sólo soporta nativamente desarrollos realizados en C++.

Generación de código

Aesop genera código C++. Aunque opera primariamente desde una interfaz visual.

Disponibilidad de plataforma

Aesop no está disponible en plataforma Windows, aunque naturalmente puede utilizarse para modelar sistemas implementados en cualquier plataforma.

1.5.3 Darwin

Darwin es un lenguaje de descripción arquitectónica. Describe un tipo de componente mediante una interfaz consistente en una colección de servicios que son ya sea provistos (declarados por ese componente) o requeridos (o sea, que se espera ocurran en el entorno). Las configuraciones se desarrollan instanciando las declaraciones de componentes y estableciendo vínculos entre ambas clases de servicios.

Objetivo principal

Como su nombre lo indica, Darwin está orientado más que nada al diseño de arquitecturas dinámicas y cambiantes.

Estilos

El soporte de Darwin para estilos arquitectónicos se limita a la descripción de configuraciones parametrizadas, es decir, se presta mejor a la descripción de sistemas que poseen características dinámicas.

Interfaces

En Darwin las interfaces de los componentes consisten en una colección de servicios que pueden ser provistos o requeridos.

Conectores

En Darwin no es posible ponerle nombre, o reutilizar un conector. Tampoco se pueden describir patrones de interacción independientemente de los componentes que interactúan.

Semántica

Darwin proporciona una semántica para sus procesos estructurales mediante el cálculo. Cada servicio se modela como un nombre de canal, y cada declaración de enlace (binding) se entiende como un proceso que trasmite el nombre de ese canal a un componente que requiere el servicio. Este modelo se ha utilizado para demostrar la corrección lógica de las configuraciones de Darwin. Dado que el cálculo ha sido designado específicamente para procesos móviles, su uso como modelo semántico confiere a las configuraciones de Darwin un carácter potencialmente dinámico. En un escenario como el Web, en el que las entidades que interactúan no están ligadas por conexiones fijas ni caracterizadas por propiedades definidas de localización, esta clase de cálculo se presenta como un formalismo extremadamente útil.

Análisis y verificación

Darwin no proporciona una base adecuada para el análisis del comportamiento de una arquitectura. Esto es debido a que el modelo no posee herramientas para describir las propiedades de un componente o de los servicios que presta. Las implementaciones de un componente vendrían a ser de este modo cajas negras no interpretadas, mientras que los tipos de servicio son una colección dependiente de la plataforma cuya semántica también se encuentra sin interpretar en el framework de Darwin.

Interfaz gráfica

Darwin proporciona notación gráfica. Existe también una herramienta gráfica (Software Architect's Assistant) que permite trabajar visualmente con lenguaje Darwin.

Soporte de lenguajes

Darwin soporta desarrollos escritos en C++, aunque no presupone que los componentes de un sistema real estén programados en algún lenguaje en particular.

Disponibilidad de plataforma

Aunque el ADL fue originalmente planeado para ambientes tan poco vinculados al modelado corporativo como hoy en día lo es Macintosh, en Windows se puede modelar en lenguaje Darwin utilizando Software Architect's Assistant.

1.5.4 Rapide

Se puede caracterizar como un lenguaje de descripción de sistemas de propósito general que permite modelar interfaces de componentes y su conducta observable. Sería tanto un ADL como un lenguaje de simulación. La estructura de Rapide es sumamente compleja, y en realidad articula cinco lenguajes: el lenguaje de tipos describe las interfaces de los componentes; el lenguaje de arquitectura describe el flujo de eventos entre componentes; el lenguaje de especificación describe restricciones abstractas para la conducta de los componentes; el lenguaje ejecutable describe módulos ejecutables; y el lenguaje de patrones describe patrones de los eventos. Los diversos sub-lenguajes comparten la misma visibilidad, scoping y reglas de denominación, así como un único modelo de ejecución.

Objetivo principal

Simulación y determinación de la conformidad de una arquitectura.

Interfaces

En Rapide los puntos de interfaz de los componentes se llaman constituyentes.

Conectores

En Rapide no es posible poner nombre, sub-tipear o reutilizar un conector.

Semántica

Mientras muchos lenguajes de tipo ADL no soportan ninguna especificación semántica de sus componentes más allá de la descripción de sus interfaces, Rapide permite modelar la conducta de sus componentes. Define tipos de componentes (llamados interfaces) en términos de una colección de eventos de comunicación que pueden ser observados (acciones externas) o iniciados (acciones públicas). Las interfaces de Rapide definen el comportamiento computacional de un componente vinculando la observación de acciones externas con la iniciación de acciones públicas.

Análisis y verificación automática

En Rapide, el monitoreo de eventos y las herramientas nativas de filtrado facilitan el análisis de arquitectura. También es posible implementar verificación de consistencia y análisis mediante simulación. En esencia, en Rapide toda la arquitectura es simulada, generando un conjunto de eventos que se supone es compatible con las especificaciones de interfaz, conducta y restricciones. La

simulación es entonces útil para detectar alternativas de ejecución. Rapide también proporciona una caja de herramientas específica para simular la arquitectura junto con la ejecución de la implementación. Sin embargo, un proceso de ejecución solamente provee una idea del comportamiento con un juego particular de variables antes que una confirmación de la conducta frente a todos los valores y escenarios posibles. Esto implica que una corrida del proceso de simulación simplemente testea la arquitectura, y no proporciona un análisis exhaustivo del escenario. Nada garantiza que no pueda surgir una inconsistencia en una ejecución diferente.

Interfaz gráfica

Rapide soporta notación gráfica.

Soporte de lenguajes

Rapide soporta construcción de sistemas ejecutables especificados en VHDL, C. C++, y Rapide mismo. Rapide no contempla estilos de la misma manera que la mayor parte de los otros ADLs.

Observaciones

En materia de evolución y soporte de sub-tipos, Rapide soporta herencia, análoga a la de los lenguajes OOP.

Disponibilidad de plataforma

Rapide ha desarrollado un conjunto de herramientas que sólo se encontraba disponible para Solaris 2.5, SunOS 4.1.3. y Linux.

1.5.5 Wright.

Se puede caracterizar como una herramienta de formalización de conexiones arquitectónicas. (Figura 4)

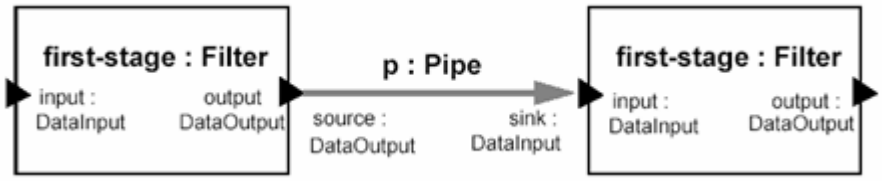

Figura 4. Diagrama correspondiente al código en Wright

Objetivo principal

Wright es probablemente la herramienta más acorde con criterios académicos de métodos formales. Su objetivo declarado es la integración de una metodología formal con una descripción arquitectónica y la aplicación de procesos formales tales como álgebras de proceso y refinamiento de procesos a una verificación automatizada de las propiedades de las arquitecturas de software.

Estilos

En Wright se introduce un vocabulario común declarando un conjunto de tipos de componentes y conectores y un conjunto de restricciones. Cada una de las restricciones declaradas por un estilo representa un predicado que debe ser satisfecho por cualquier configuración de la que se declare que es miembro del estilo. La notación para las restricciones se basa en el cálculo de predicados de primer orden. Las restricciones se pueden referir a los conjuntos de componentes, conectores y attachments, a las puertos y a las computaciones de un componente específico y a los roles y ligamentos (glue) de un conector particular. Es asimismo posible definir sub-estilos que heredan todas las restricciones de los estilos de los que derivan. No existe, sin embargo, una manera de verificar la conformidad de una configuración a un estilo canónico estándar.

Interfaces

En Wright los puntos de interfaz se llaman puertos (ports).

Semántica

Mientras muchos lenguajes de tipo ADL no soportan ninguna especificación semántica de sus componentes más allá de la descripción de sus interfaces, Wright y Rapide permiten modelar la conducta de sus componentes. En Wright existe una sección especial de la especificación llamada Computation que describe la funcionalidad de los componentes.

Análisis y verificación automática

Wright permite analizar los conectores para verificar que no haya deadlocks. Wright define verificaciones de consistencia que se aplican estáticamente, y no mediante simulación. Esto lo hace más confiable que Rapide. También se puede especificar una forma de compatibilidad entre un puerto y un rol, de modo que se pueda cambiar el proceso de un puerto por el proceso de un rol sin que la interacción del conector detecte que algo ha sido modificado. Esto permite implementar relaciones de refinamiento entre procesos. En Wright existen además recaudos (basados en teoremas formales relacionados con CSP) que garantizan que un conector esté libre de deadlocks en todos los escenarios posibles.

Interfaz gráfica

Wright no proporciona notación gráfica en su implementación nativa.

Generación de código

Wright se considera como una notación de modelado autocontenida y no genera (al menos en forma directa) ninguna clase de código ejecutable.

Disponibilidad de plataforma

Wright es en principio independiente de plataforma y es un lenguaje formal, antes que una herramienta paquetizada. Debido a que se basa en CSP (Communicating Sequential Process), cualquier solución que pueda tratar código CSP en plataforma Windows es apta para obtener ya sea una visualización del modelo o verificar la ausencia de deadlocks o la consistencia del modelo. El código susceptible de ser manejado por herramientas de CSP académicas o comerciales requiere tratamiento previo por un módulo de Wright que por el momento existe sólo para Linux o SunOS; pero una vez que se ha generado el CSP (lo cual puede hacerse manualmente insertando unos pocos tags) se puede tratar en Windows con cualquier solución ligada a ese lenguaje, ya sea FDR o alguna otra.

1.6 Estilos arquitectónicos.

La clave del trabajo arquitectónico tiene que ver con la correcta elección del estilo arquitectónico.

¿Qué es un estilo arquitectónico?

En el caso de los “estilos arquitectónicos” de software son arquitecturas de software comunes, marcos de referencias arquitectónicas, formas comunes o clases de sistemas.
Los estilos de arquitectura se definen como las 4C [3]:
Componentes (Elementos)
Conectores
Configuraciones
Restricciones (Constraints)
A la hora de definir un estilo arquitectónico es necesario tener en cuenta el tipo de aplicación ya que puede imponer restricciones que acotan la interacción de los componentes, además se tiene en cuenta el patrón de organización general.

Estilos arquitectónicos
Arquitectura basada en eventos
Tubería y Filtros
Modelo-Vista-Controlador
Pizarra
Máquina virtual
Arquitectura basada en objetos
Arquitectura en capas
C2
Modelo REST

1.6.1 Arquitectura basada en eventos

- Impiden incurrir en el modelo de aplicaciones que preguntan si sucedió algo.
- Generan la ejecución apenas ocurre el evento o el usuario se conecta.
- Modelo de push. A veces se vincula con patrón Observador (Observer pattern).

Ventajas
- Simplicidad – Fácil programación.
- Evolución: se pueden reemplazar componentes suscriptores.
- Modularidad: una sola modalidad para eventos diversos.

- Puede mejorar eficiencia, eliminando la necesidad de polling por ocurrencia de evento.

Desventajas

- Posibilidad de desborde.
- Potencial imprevisión de escalabilidad.
- Pobre comprensibilidad: Puede ser difícil prever qué pasará en respuesta a una acción.
- No hay garantía del lado del publisher que el suscriptor responderá al evento.
- No hay mucho soporte de recuperación en caso de falla parcial.

1.6.2 Tubería y filtros

Una tubería (pipeline) es una popular arquitectura que conecta componentes computacionales (filtros) a través de conectores (pipes), de modo que las computaciones se ejecutan a la manera de un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas (Figura 5). Debido a su simplicidad y su facilidad para captar una funcionalidad, es una arquitectura mascota cada vez que se trata de demostrar ideas sobre la formalización del espacio de diseño arquitectónico, igual que el tipo de datos stack lo fue en las especificaciones algebraicas o en los tipos de datos abstractos.

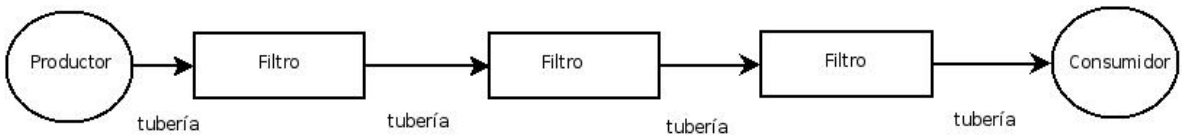

Figura 5. Estructura

Un paso en un sistema T-F consiste en una de dos cosas:

- Una transformación incremental de los datos por un filtro.
- Una comunicación de datos entre puertos por un tubo.
- Los filtros pueden descomponerse ulteriormente, los tubos no.
- Los datos no se alteran durante su trasmisión.
- El orden de los datos es el mismo cuando sale de un filtro que cuando entra al siguiente.

- Los tubos crean el contexto en el que operan los filtros, pero no operan independientemente de ellos.
- Los tubos conectan exactamente dos puertos.
- Los “filtros” en realidad realizan otras operaciones; sólo algunas veces son de filtrado.

Ventajas

- Es simple de entender e implementar. Es posible implementar procesos complejos con editores gráficos de líneas de tuberías o con comandos de línea por ejemplo, transformaciones mediante Mapper de XSLT (BizTalk).
- Lineal: la conducta del sistema es la suma de la conducta de los sucesivos filtros.
- Fuerza un procesamiento secuencial.
- Es fácil de envolver (wrap) en una transacción atómica.
- Los filtros se pueden empaquetar, y hacer paralelos o distribuidos.

Desventajas

- El patrón puede resultar demasiado simplista, especialmente para orquestación de servicios que podrían ramificar la ejecución de la lógica de negocios de formas complicadas.
- No maneja con demasiada eficiencia construcciones condicionales, bucles y otras lógicas de control de flujo. Agregar un paso suplementario afecta la performance de cada ejecución de la tubería.
- No es posible que 2 o más filtros cooperen.
- Eventualmente pueden llegar a requerirse buffers de tamaño indefinido, por ejemplo en las tuberías de clasificación de datos.
- No es apto para manejar situaciones interactivas, sobre todo cuando se requieren actualizaciones incrementales de la representación en pantalla.
- La independencia de los filtros implica que es muy posible la duplicación de funciones de preparación que son efectuadas por otros filtros (p. ej. el control de corrección de un objeto de fecha).
- Es complicado manejar uniformemente gestión de errores en filtros diferentes.
- Generalmente fuerzan mínimo común denominador en representación de datos (texto ASCII). Si se deben procesar datos tokenizados de otra manera, se requieren parsers y serializadores específicos.

1.6.3 Pizarra

- Cuándo se utiliza: Problemas no susceptibles de tratarse analíticamente.
- Reconocimiento de patrones, aprendizaje de máquina, data mining.
- Dos formas:
- Repositorio.
- Pizarra pura o tablero de control.
- Procesamiento de señales.
- Reconocimiento de habla.
- Redes neuronales.
- Agentes autónomos (débilmente acoplados).

Variante repositorio

• Base de datos o almacenamiento persistente.

• Es pasivo.

• Los clientes accedan a datos compartidos cuando lo requieren.

Variante pizarra propiamente dicha

• Es activo.

• Pueden enviar notificación a suscriptores cuando la información cambia.

• El estado es el disparador de la acción a ejecutar.

Variante: Repositorio

Ventajas

• Forma adecuada de manejar grandes volúmenes de datos.

• Administración centralizada.

Desventajas

• Los subsistemas tienen que estar de acuerdo en el modelo de datos del repositorio.

• La evolución de los datos es difícil y cara.

• Difícil de distribuir con facilidad.

1.6.4 Arquitectura en capas

• Resulta útil cuando se pueden identificar distintas clases de servicios que pueden ser articulados jerárquicamente.
• Puede verse como una jerarquía de máquinas virtuales cada vez más poderosas y abstractas.
• Los componentes de cada capa consisten en conjuntos de procedimientos.
• Las interacciones entre capas usualmente proceden por invocación de procedimientos.
• Por definición, los niveles de abajo no pueden usar funcionalidad ofrecida por los de arriba.
• Problema de diseño: cómo articular la funcionalidad de cada capa

Ventajas
• Modularidad (hasta cierto punto).

Desventajas
• Es difícil razonar a priori sobre la separación en capas requerida.
• Capas disjuntas requieren drivers de otras capas.
• Formatos, protocolos y transportes de la comunicación entre capas suelen ser específicos y propietarios.
• Algunos componentes pueden estar lógicamente vinculados con más de una capa.
• En ocasiones, hay que pasar por encima de capas intermedias.
• Otras veces, razones de performance hace que se sitúen funciones en las capas indebidas (lógica de negocios en procedimientos almacenados).
• La multiplicación de capas genera procesos adicionales de comunicación y coordinación.

1.6.5 Etapas en la confección de un software educativo

En la literatura especializada en el tema existe un sin número de metodologías que tienen por objetivo desarrollar software de calidad. Vamos a analizar una que nos parece apropiada, sencilla y que permite llevar a buen término un proyecto de software educativo.

Esta metodología tiene su base en el denominado modelo de cascada y resume diferentes aspectos relacionados con el diseño de materiales instruccionales. Consta de cinco fases o etapas: análisis y requerimientos, diseño, construcción, prueba y mantenimiento. El modelo se denomina de cascada puesto que se supone se parte del tope y se va pasando a la fase siguiente cuando la anterior está cumplida con determinado rigor. A continuación, se ofrece una breve explicación de cada una de ellas:

Análisis y requerimientos

Resulta de vital importancia, ya que en esta etapa se realiza una descripción detallada del objeto de estudio, y se elaboran todas las especificaciones, tanto las que se relacionan con la construcción, como con el uso del software.

En esta etapa debe quedarnos claro entre otras cuestiones: la necesidad de elaborar el producto (problema pedagógico a resolver), el público al que va dirigido, los objetivos pedagógicos que se pretenden cumplir, los contenidos a tratar y los medios para presentarlos, las herramientas que se utilizarán para el desarrollo, los equipos de trabajo que se conformarán, el hardware necesario tanto para realizadores como para usuarios, la factibilidad técnica y económica de su producción (presupuesto necesario), las formas de distribución y la primera versión del cronograma de trabajo. El resultado más significativo de esta etapa es la escritura de la primera versión del guión.

Diseño

En esta etapa se obtendrá una información detallada de cómo estará estructurado el programa, cómo progresa o fluye a través de cualquier opción posible dentro de él, elegida por el usuario o por la computadora. Debe incluir, por tanto, un análisis de modularidad y jerarquía (la utilización de mapas conceptuales favorece el trabajo), y tener en cuenta todos los requerimientos del público al que está dirigido y ante todo el diseño de la interfaz de cada una de las pantallas. Aquí se define la organización interna del producto (directorios, archivos, etc.). También debe quedar definido el protocolo de pruebas que se empleará. Lamentablemente, en muchas ocasiones no se pone suficiente atención a esta etapa que nos debe brindar una versión acabada del guión (aunque sujeta a cambios en función de los resultados que arrojen las pruebas) y, como resultado, la programación es menos organizada, toma más tiempo y lo que es peor aún, la revisión se hace más difícil y en ocasiones imposible.

Construcción

Aquí se cumplen dos tareas de singular importancia: la obtención y edición de todos los medios que serán empleados y la programación, es decir, la codificación de los módulos definidos con anterioridad. Al final de esta fase se debe tener un código claro y documentado, así como trabajar por la utilización de herramientas y bibliotecas comunes. Si además de los sistemas de ayuda que se programen para asistir al usuario durante la ejecución del software se decide incluir otra documentación, manual o recomendaciones, deben ser escritas en esta etapa.

Prueba

Es necesaria una comprobación sistemática para buscar los posibles errores; se debe velar por el cumplimiento satisfactorio de los objetivos relacionados con la confiabilidad del software desde los puntos de vista conceptual, de la utilización y de la representación o codificación. En la evaluación sistemática del prototipo y del producto final, deben participar no sólo el colectivo de realización sino también otros expertos en informática educativa y de la materia en cuestión, además de una representación del público al que está dirigido el software.

Mantenimiento

La correcta utilización de una metodología en el desarrollo de un software, posibilita el mantenimiento efectivo de éste. Se hace necesario actualizar los comentarios del código y la documentación correspondiente para hacer cualquier modificación que garantice la competitividad del producto. Un registro de usuarios permite obtener, de forma real, un análisis riguroso de dificultades y errores en el software, así como de sus aciertos.

1.7 Relación existe entre Estilos y Patrones.

Una vez que se ha decidido que estilos se van a usar en la aplicación los patrones que tienen relación con estos estilos ayudarán a derivar de esa formulación arquitectónica el diseño y ulteriormente la programación correspondiente. Los estilos sólo se manifiestan en arquitectura teórica descriptiva de alto nivel de abstracción; los patrones, por todas partes.

1.7.1 ¿Qué son los Patrones?

Los patrones fueron sugeridos por Christopher Alexander en 1977, el cual escribió una serie de libros y artículos referentes a ¿qué son los patrones?, los textos más frecuentemente encontrados se refieren a patrones arquitectónicos en cuanto a arquitectura real, o sea de edificios y ciudades y no de arquitectura de aplicaciones de software.

Una de las ideas brillantes de la década de los 90 fue vincular las estructuras recurrentes, aquellas entidades que ocurrían una y otra vez; las soluciones recurrentes a problemas en determinados contextos con esta idea de Alexander que estaba referida como se dijo anteriormente a la arquitectura real.

• Un patrón es la solución a un problema en un contexto.
• Un patrón codifica conocimiento específico acumulado por la experiencia en un dominio.
• Un sistema bien estructurado está lleno de patrones.
• “Cada patrón describe un problema que ocurre una y otra vez en nuestro ambiente, y luego describe el núcleo de la solución a este problema, de tal manera que puede usar esa solución un millón de veces más, sin hacer jamás la misma cosa dos veces”.

1.7.1.2 Los patrones pueden dividirse o clasificarse en:

Patrones de Arquitectura: Relacionados a la interacción de objetos dentro o entre niveles arquitectónicos, problemas arquitectónicos, adaptabilidad a requerimientos cambiantes, performance, modularidad, acoplamiento.

Patrones de Diseño: que fueron construidos dado los problemas con la claridad de diseño, multiplicación de clases, adaptabilidad a requerimientos cambiantes, proponiendo como solución: Comportamiento de factoría, Clase-Responsabilidad-Contrato (CRC).

Patrones de Análisis: Usualmente específicos de aplicación. Patrones de Proceso o de Organización: que tratan todo lo concerniente con el desarrollo o procesos de administración de proyectos, o técnicas, o estructuras de organización.

Patrones de Idioma: que reglan la nomenclatura en la cual se escriben, se diseñan y desarrollan nuestros sistemas.

1.7.1.3 Los elementos de un patrón son:

Nombre

Define un vocabulario de diseño.

Facilita la abstracción.

Problema

Describe cuándo aplicar el patrón.

Conjunto de fuerzas: objetivas y restricciones.

Prerrequisitos.

Solución

Elementos que constituyen el diseño (plantilla).

Forma canónica para resolver fuerzas.

Consecuencias

Resultados.

Extensiones.

Consensos.

Al contrario de los estilos arquitectónicos los patrones son muchos y a su vez muy variados y es casi imposible revisar todos los patrones que existen a la hora de hacer una determinada aplicación, por eso se recomienda el uso de los patrones que estén asociados en cada uno de los estilos que se seleccionen para el desarrollo de la arquitectura.

1.7.2 ¿Qué son los estilos?

El tópico más urgente y exitoso en arquitectura de software en los últimos cuatro o cinco años es, sin duda, el de los patrones (patterns), tanto en lo que concierne a los patrones de diseño como a los de arquitectura. Inmediatamente después, en una relación a veces de complementariedad, otras de oposición, se encuentra la sistematización de los llamados estilos arquitectónicos.

Los patrones coronan una práctica de diseño que se origina antes que la arquitectura de software se distinguiera como discurso en perpetuo estado de formación y proclamara su independencia de la ingeniería en general y el modelado en particular. Los estilos, en cambio, expresan la arquitectura en el sentido más formal y teórico, constituyendo un tópico esencial de lo que Goguen ha llamado el campo "seco" de la disciplina.

Los estilos se encuentran en el centro de la arquitectura y constituyen buena parte de su sustancia. Los patrones de arquitectura están claramente dentro de la disciplina arquitectónica, solapándose con los estilos, mientras que los patrones de diseño se encuentran más bien en la periferia, si es que no decididamente afuera.

Un estilo arquitectónico encapsula decisiones esenciales sobre los elementos arquitectónicos y enfatiza restricciones importantes de los elementos y sus relaciones posibles.

1.7.2.1 Clases de Estilos

Existe una sistematización de clases de estilos que se dividen en seis grupos:

1. Flujo de datos
- Secuencial en lotes.
- Red de flujo de datos (tuberías y filtros).
- Bucle de control cerrado.

2. Llamada y Retorno
- Programa principal / subrutinas.
- Ocultamiento de información (ADT, objeto, cliente/servidor elemental).

3. Procesos interactivos
- Procesos comunicantes.

- Sistemas de eventos (invocación implícita, eventos puros).

4. Repositorio Orientado a Datos
- Bases de datos transaccionales (cliente/servidor genuino).
- Pizarra.
- Compilador moderno.

5. Datos Compartidos
- Documentos compuestos.
- Hipertexto.
- Fortran COMMON.
- Procesos LW.

6. Jerárquicos
- En capas (intérpretes).

Los estilos casi siempre se usan combinados, cada capa o componente puede ser internamente de un estilo diferente al de la totalidad. Muchos estilos se encuentran ligados a dominios específicos, o a líneas de producto particulares.

1.8 Conclusiones

A lo largo de este capítulo se han ofrecido los elementos teóricos que sirven de sustento científico a la investigación, así como cada uno de los aspectos a tener en cuenta en la situación problemática que genera el problema científico razón de esta investigación. El análisis de cada uno de estos aspectos mencionados permite arribar de manera parcial a las siguientes conclusiones:

- Para la producción de software educativo, en los últimos tiempos se ha incrementado mucho la utilización de la tecnología multimedia, ya que brinda muchas ventajas para lograr los objetivos perseguidos en un software educativo.

- Las buenas prácticas de la arquitectura de software aplicadas durante el ciclo de vida, tienen el potencial de incrementar la facilidad para entender un sistema de software y de desarrollar

procesos para crearlo, asegurando qué cualidades particulares y relevantes sean alcanzadas, y que se reduzca el costo.

Capítulo 2: “Propuesta de Solución”

2.1 Introducción

Las necesidades actuales para el logro de los objetivos el proceso de creación de software educativo, demandan la construcción de grandes y complejos sistemas de software que requieren de la combinación de diferentes tecnologías y plataformas de hardware y software para alcanzar un funcionamiento acorde con dichas necesidades. Lo anterior, exige de los profesionales dedicados al desarrollo de software poner especial atención y cuidado al diseño de la arquitectura, bajo la cual estará soportado el funcionamiento de sus sistemas.

Si no contamos con una arquitectura de software del sistema que desarrollamos, o se encuentra deficiente en su concepto o diseño, tendremos grandes posibilidades de construir un sistema que no alcanzará el total de los requerimientos establecidos.

De esta manera, es necesario conocer y comprender los elementos que deben evaluarse al diseñar una arquitectura de software.

2.2 Modelo de desarrollo del software educativo

El rendimiento del sistema educativo actual y los vertiginosos adelantos de la ciencia y la tecnología hacen evidente la urgente necesidad de introducir innovaciones metodológicas, técnicas, empleo de medios y recursos complementarios a la educación tradicional, con el fin de mejorar el aprendizaje en la población estudiantil.

2.2.1 ¿Qué es un modelo de desarrollo?

En términos generales se puede afirmar que un modelo es una representación simplificada de los fenómenos del mundo real que da sentido a esta realidad, de modo que se puedan comprender las situaciones complejas y podamos hacer predicciones. Frente a esta definición, es necesario por un lado, establecer que para construir un modelo pertinente y coherente, es necesario haber comprendido

los aspectos esenciales de la realidad del objeto estudiado, y por otro lado, establecer diferencias entre los modelos estáticos, que son las representaciones de una existencia cualquiera, de los modelos dinámicos, cuyo funcionamiento permite simular procesos.

El planteamiento de un modelo de desarrollo de software educativo, implica inexpugnablemente, la construcción de conocimiento no sólo del desarrollo, sino también sobre su relación con otros procesos como el diseño y evaluación, es decir, el modelo se plantea para generar un conocimiento acerca del proceso de desarrollo del software educativo, pero a su vez, su construcción requiere de un conocimiento de este y de otros procesos.

En este caso, el modelo de desarrollo de software educativo, implica el planteamiento de los diferentes aspectos que componen este proceso, de las personas que participan en él y de los procedimientos y etapas que lo componen. Cada uno de estos elementos se presenta, además, con las relaciones que se pueden establecer en ellos y las diferentes alternativas que se pueden utilizar para su representación.

Son múltiples las propuestas que se encuentran para la producción de software educativo, la mayoría de ellas dirigidas al desarrollo de software prototípico, elaborado por grupos especializados de profesionales.

2.2.2 Múltiples modelos de desarrollo de software

Para el planteamiento del modelo del proceso de desarrollo de software educativo, es necesario tener en consideración los siguientes principios fundamentales:

- Un modelo es necesariamente la simplificación de la realidad, si no sería la realidad misma y no contribuiría en nada a la construcción del conocimiento sobre el proceso de desarrollo de software educativo. Así, es una representación abstracta, lo cual significa que la aproximación -e incluso el error- es inherente al modelo.
- El de desarrollo de software educativo es un proceso complejo, en el que entran en juego múltiples factores de carácter cultural, social, educativo, técnico y comunicativo, que lo convierten en un proceso dinámico, es decir, en construcción y evolución permanente.
- No se puede afirmar que exista un solo tipo de software educativo, sino que este ha venido evolucionando de acuerdo con las innovaciones tecnológicas que han redundado en la existencia de

una variada gama de software educativo, que van desde los software de tipo transmisionista, pasando por el software activo, hasta llegar al interactivo.

- En el proceso de desarrollo de software educativo es necesario tener en cuenta los procesos educativos que apoyan y que se fundamentan en enfoques pedagógicos variados, de acuerdo con las diferentes tendencias utilizadas, las cuales van desde enfoques instruccionalistas, tradicionales, pasando por enfoques activos, críticos, constructivistas e interaccionistas.
- El desarrollo de software educativo se apoya en procesos técnicos para su producción, los cuales, al igual que los enfoques pedagógicos, son variados y dependen de las plataformas utilizadas, el tipo de tecnología disponible y aquella que se pueda utilizar en el sector educativo.

2.3 Criterios de desarrollo de software

Las anteriores consideraciones ponen de presente la necesidad de tener en cuenta el contexto de desarrollo del software educativo y por lo tanto, la posibilidad de plantear más de un modelo, de tal manera que se responda a la variedad de tipos de software utilizado, los diferentes enfoques pedagógicos, herramientas y plataformas técnicas que se pueden utilizar. Por ello, a continuación, se plantean los criterios generales y actores a tener en cuenta en los modelos de desarrollo y finalmente, algunas consideraciones puntuales sobre el proceso mismo de desarrollo. (Tabla #1)

Tabla 1. Criterios de desarrollo de software educativo

TIPO DE MODELO CRITERIOS	SOFTWARE PROTOTIPO	SOFTWARE ESTRUCTURADO	SOFTWARE EVOLUTIVO
Necesidad que lo origina	Necesidades específicas de aprendizaje.	Distancias geográficas y de tiempo.	Compartir información para construir conocimiento.
Componentes del modelo	Pedagógico, Tecnológico, Disciplinar, Gráfico.	Pedagógico, Tecnológico, Disciplinar, Comunicativo, Administrativo.	Multicomponente. Sin definir.
Formas de presentación de información	Medios expositivos Medios activos.	Medios expositivos, activos e interactivos.	Medios activos. Medios interactivos.
Plataformas utilizadas	Software propietario. Software libre.	Software propietario. Software libre.	Software libre.
Tipo de producto	CD Multmedial	Objetos y cursos virtuales	Blogs personales, escritos colaborativos.
Forma de desarrollo	Tipo catedral	Tipo catedral	Tipo bazar
Equipo de Desarrollo	Desarrollador independiente Equipo de Producción.	Equipo de Producción.	Equipo abierto.
Proceso de	Por proyecto.	Por proyecto.	Colaborativo

Desarrollo			
Tipo de financiación	Compromiso individual. Institucional.	Institucional.	Compromiso individual.
Medio de difusión	CD	Internet	Internet
Tipo de contenido	Cerrado (Contenidos protegidos por Derechos de autor y Derechos de comercialización).	Cerrado (Contenidos protegidos por Derechos de autor y Derechos de comercialización). Abierto (Público, multiautor)	Abierto (Público, multiautor)
Usuarios del modelo	Docentes, especialistas, estudiantes de primaria, secundaria, media, universitaria)	Estudiantes de media, universitarios de pregrado y postgrados, educación permanente y no formal.	Todo tipo de persona.
Usos metodológicos.	Grupos formales.	Redes de aprendizaje.	Comunidades virtuales.

El proceso de desarrollo de software educativo, se fundamenta en unos criterios generales que orientan el proceso y permiten establecer diferencias entre, al menos, tres categorías de software educativo. Estos criterios de pueden dividir entre aquellos encaminados al proceso mismo de desarrollo y otros dirigidos a determinar las características de los actores que intervienen en el modelo. Dentro de los primeros se encuentran los siguientes:

2.4 Notaciones en la producción del Software Educativo Cubano

El guión

"La función primordial del guión es dar objetividad al proyecto de la obra multimedia de forma tal que pueda independizarse el proceso de ejecución del proyecto de concepción y diseño. (…) En segundo lugar, el guión es imprescindible para lograr una comunicación clara y precisa entre los integrantes del equipo de trabajo de manera que aunque realicen tareas independientes todos conozcan cómo tienen que hacer su labor y cómo encaja cada uno dentro de la obra." (Barrera Yanes, 1998)

Otros autores hacen una clasificación más sencilla siguiendo el esquema clásico de las producciones audiovisuales. A diferencia de éstas, en el guión multimedia:

"(…) se recogerán de manera exhaustiva los elementos que han de intervenir en cada una de las pantallas o secuencias, las acciones que se desarrollarán, el grafismo utilizado y el "tempo" que mantendrán, es decir, el orden de representación y el tiempo del mismo. También se debe describir

detalladamente cuándo sucederá, bien por la intervención directa del usuario bien por otro tipo de causas, como por ejemplo la ausencia de acciones por parte de éste". (Pérez Huertas, 1998)

Árboles o mapas de navegación.

Un mapa de navegación es la representación gráfica de la organización de la información. Expresa todas las relaciones de jerarquía y secuencia y permite elaborar escenarios de comportamiento de los usuarios. También grafica, de modo que todos los profesionales participantes en un proyecto lo tengan claro, diferencias entre páginas dinámicas, administrables o estáticas.

El principal valor de un mapa de navegación es que permite anticipar errores de organización de la información, de modo de corregirlos cuando aún no se ha invertido tiempo y dinero en la construcción del producto.

2.5 Funcionalidades del Software Educativo

Los programas didácticos cuando se aplican a la realidad educativa, realizan las funciones básicas propias de los medios didácticos en general. (Marqués 1996)

No se puede afirmar que el software educativo por si mismo sea bueno o malo, todo dependerá del uso que se le de, y de la manera en que se utilice en cada situación concreta. Por último la funcionalidad, las ventajas e inconvenientes que pueda tener el uso de los materiales didácticos, será el resultado de las características del material, de su adecuación al contexto educativo al que se aplica y de la manera en que el profesor organice su utilización (si el profesor no organiza bien su utilización de manera que se logre los objetivos que con el software se pretende, los cuales deben ser similares a los objetivos pedagógicos del contexto al que se aplica, el software no cumple con los requisitos funcionales explícitamente establecidos).

Entre los elementos funcionales del software educativo podemos citar:

Originalidad y uso de la tecnología avanzada: Los programas didácticos deben presentar entornos originales, bien diferenciados de otros materiales didácticos y utilizar las crecientes potencialidades del ordenador y de las tecnologías multimedia e hipertexto en general, de forma tal que el ordenador resulte intrínsecamente potenciador del proceso de aprendizaje, favorezca la asociación de ideas y creatividad, permita la practica de nuevas técnicas, la reducción del tiempo y del esfuerzo necesario para aprender y facilite aprendizajes más completos y significativos. La inversión financiera, intelectual

y metodológica que supone elaborar un programa educativo, solo se justifica si el ordenador mejora lo que ya existe.

Facilidad de uso e instalación: Los programas educativos deben ganarse la atención de los estudiantes, para esto es necesario que sean agradables, fáciles de usar y autoexplicativos, de forma tal que los estudiantes puedan utilizarlos sin tener que pasar por un curso o tener que realizar una exhaustiva lectura de los manuales para aprender a trabajar con el software. Agreguemos a esto, que el software no debe tener largas tareas previas de configuración y además, debe posibilitarle al estudiante conocer en todo momento el lugar del programa donde se encuentra y las opciones a su alcance para moverse según sus preferencias (retroceder, avanzar o ir al índice).

Adaptación a diversos contextos: Debido a que los materiales didácticos deben dar una buena respuesta a las diversas necesidades educativas de sus destinatarios y puedan utilizarse de múltiples maneras según las circunstancias, es conveniente que tengan una alta capacidad de adaptación a:

- *Entornos de uso***:** Pueden ser aulas de informática, clases con un único ordenador, clases con pizarra electrónica (en un futuro).
- *Agrupamientos***:** Por trabajo individual, grupo cooperativo o competitivo.
- *Estrategias didácticas***:** Ya sea enseñanza dirigida, exploración guiada, y libre descubrimiento.
- *Usuarios y contextos formativos***:** Estilos de aprendizajes, circunstancias culturales y necesidades formativas, problemáticas para el acceso a la información (visual y matriz).

Para lograr esta adaptación, los materiales didácticos en soporte informático deberán tener ciertas características que lo permita, como las que siguen a continuación:

- *Programables,* para poder modificar algunos parámetros como: nivel de dificultad, tiempo de respuesta, números de usuarios simultáneos e idioma.
- *Abiertos,* permitiéndole al profesorado modificar fácilmente los contenidos de las bases de datos y las actividades que proporcionarán a los estudiantes.
- *Facilitar la impresión de los contenidos***,** sin una excesiva fragmentación. .. Incluir un sistema de evaluación y seguimiento (control), que proporcione informes de las actividades realizadas por los estudiantes: temas tratados, nivel de dificultad, tiempo invertido, errores que ha cometido, itinerarios seguidos para resolver los problemas y otros.
- *Permitir* el seguimiento de los trabajos empezados con anterioridad.

- *Promover* el desarrollo de actividades complementarias (individuales o en grupo) con el uso de otros materiales (fichas, diccionarios o libros).
- *Dar respuesta* a las problemáticas de acceso de los colectivos, con necesidades especiales, proporcionando interfaces ajustables según las características de los usuarios (tamaño de letra, uso del teclado, ratón o periféricos adaptativos).

2.6 Tipologías de los Programas Didácticos

Los programas didácticos, aunque presentan rasgos esenciales básicos y una estructura general común, se presentan de diversas formas. Con el fin de establecer un orden a esta disparidad se han elaborados múltiples tipologías que clasifican los programas didácticos a partir de diferentes criterios. Unos de estos criterios se basa en la consideración del tratamiento de los errores que cometen los estudiantes, distinguiéndose (Marqués 1996).

- Programas de tutoriales directivos**,** que hacen preguntas a los estudiantes y controlan en todo momento su actividad. El ordenador adopta el papel de juez poseedor de la verdad y examina al alumno. Se producen errores cuando la respuesta del alumno está en desacuerdo con la que el ordenador tiene como correcta. El error lleva implícita la noción del fracaso.
- *Programas no directivos***,** en los que el ordenador adopta el papel de un laboratorio o instrumento a disposición de la iniciativa de un alumno que pregunta y tiene una libertad de acción sólo limitada por las normas del programa. El ordenador no juzga las acciones del alumno, se limita a procesar los datos que éste introduce y a mostrar las consecuencias de sus acciones sobre un entorno. Objetivamente no se producen errores, sólo desacuerdos entre los efectos esperados por el alumno y los efectos reales de sus acciones sobre el entorno. No está implícita la noción de fracaso. El error es sencillamente una hipótesis de trabajo que no se ha verificado y que se debe sustituir por otra. En general, siguen un modelo pedagógico de inspiración cognitivista, potencian el aprendizaje a través de la exploración, favorecen la reflexión y el pensamiento crítico y propician la utilización del método científico.

Otra clasificación interesante de los programas tiene en cuenta la posibilidad de modificar los contenidos del programa y distingue entre programas cerrados (que no pueden modificarse) y abiertos (que pueden modificarse), que proporcionan un esqueleto, una estructura, sobre la cual los alumnos y

los profesores pueden añadir el contenido que les interese. De esta manera se facilita su adecuación a los diversos contextos educativos y permite un mejor tratamiento de la diversidad de los estudiantes.
Pero la clasificación que tiene en cuenta el grado de control del programa sobre las actividades de los alumnos y la estructura de su algoritmo, es la que proporciona categorías más claras y útiles a los profesores, por ende, es la que se procede a dar una breve explicación a continuación.

Programas Tutoriales

Son programas que en mayor o menor medida dirigen y tutorizan, el trabajo de los alumnos. Pretenden que, a partir de unas informaciones y mediante la realización de ciertas actividades previstas de antemano, los estudiantes pongan en juego determinadas capacidades y aprendan o refuercen unos conocimientos y/o habilidades. Cuando se limitan a proponer ejercicios de refuerzo sin proporcionar explicaciones conceptuales previas, se denominan programas tutoriales de ejercitación, como es el caso de los programas de preguntas (drill&practice, test) y de los programas de adiestramiento psicomotor, que desarrollan la coordinación neuromotriz en actividades relacionadas con el dibujo, la escritura y otras habilidades psicomotrices.

En cualquier caso, son programas basados en los planteamientos conductistas de la enseñanza, que comparan las respuestas de los alumnos con los patrones que tienen como correctos, guían el aprendizaje de los estudiantes y facilitan la realización de prácticas más o menos rutinarias y su evaluación; en algunos casos una evaluación negativa genera una nueva serie de ejercicios de repaso.
A partir de la estructura del algoritmo de los programas, se distinguen cuatro categorías:

- Programas lineales**,** que presentan al alumno una secuencia de información y/o ejercicios (siempre la misma o determinada aleatoriamente) con independencia de la corrección o incorrección de sus respuestas. Herederos de la enseñanza programada, transforman el ordenador en una máquina de enseñar transmisora de conocimientos y adiestradora de habilidades. No obstante, su interactividad resulta pobre y el programa se hace largo de recorrer.
- Programas ramificados**,** basados inicialmente también en modelos conductistas, siguen recorridos pedagógicos diferentes según el juicio que hace el ordenador sobre la corrección de las respuestas de los alumnos o según su decisión de profundizar más en ciertos temas. Ofrecen mayor interacción, más opciones, pero la organización de la materia suele estar menos compartimentada que en los programas lineales y exigen un esfuerzo más grande al alumno. Pertenecen a éste

grupo los programas multinivel, que estructuran los contenidos en niveles de dificultad y previenen diversos caminos, y los programas ramificados con dientes de sierra, que establecen una diferenciación entre los conceptos y las preguntas de profundización, que son opcionales.

- Entornos tutoriales, en general, están inspirados en modelos pedagógicos cognitivistas, y proporcionan a los alumnos una serie de herramientas de búsqueda y de proceso de la información que pueden utilizar libremente para construir la respuesta a las preguntas del programa. Este es el caso de los entornos de resolución de problemas, "problem solving", donde los estudiantes conocen parcialmente las informaciones necesarias para su resolución y han de buscar la información que falta y aplicar reglas, leyes y operaciones para encontrar la solución. En algunos casos, el programa no sólo comprueba la corrección del resultado, sino que también, tiene en cuenta la idoneidad del camino que se ha seguido en la resolución. Sin llegar a estos niveles de análisis de las respuestas, podemos citar como ejemplo de entorno de resolución de problemas el programa MICROLAB DE ELECTRÓNICA.
- Sistemas tutoriales expertos, como los Sistemas Tutores Inteligentes (Intelligent Tutoring Systems), que, elaborados con las técnicas de la Inteligencia Artificial y teniendo en cuenta las teorías cognitivas sobre el aprendizaje, tienden a reproducir un diálogo auténtico entre el programa y el estudiante, y pretenden comportarse como lo haría un tutor humano: guían a los alumnos paso a paso en su proceso de aprendizaje, analizan su estilo de aprender y sus errores y proporcionan en cada caso la explicación o ejercicio más conveniente.

Base de Datos**:**

Proporcionan datos organizados, en un entorno estático, según determinados criterios, y facilitan su exploración y **consulta** selectiva. Se pueden emplear en múltiples actividades como por ejemplo: seleccionar datos relevantes para resolver problemas, analizar y relacionar datos, extraer conclusiones, y comprobar hipótesis. Las preguntas que acostumbran a realizar los alumnos son del tipo: **¿Qué características tiene este dato? ¿Qué datos hay con la característica X? ¿Qué datos hay con las características X y Y?**

Las bases de datos pueden tener una estructura jerárquica (si existen unos elementos subordinantes de los que dependen otros subordinados, como los organigramas), relacional (si están organizadas mediante unas fichas o registros con una misma estructura y rango) o documental (si utiliza descriptores y su finalidad es almacenar grandes volúmenes de información documental: revistas, periódicos). En cualquier caso, según la forma de acceder a la información se pueden distinguir dos tipos:

- *Bases de datos convencionales*, tienen la información almacenada en ficheros, mapas o gráficos, que el usuario puede recorrer según su criterio para recopilar información.
- *Bases de datos tipo sistema experto*, son bases de datos muy especializadas que recopilan toda la información existente de un tema concreto y además asesoran al usuario cuando accede buscando determinadas respuestas.

Simuladores:

Presentan un modelo o entorno dinámico (generalmente a través de gráficos o animaciones interactivas), facilitan su exploración y modificación a los alumnos, que pueden realizar aprendizajes inductivos o deductivos, mediante la observación y la manipulación de la estructura subyacente; de esta manera pueden descubrir los elementos del modelo, sus interrelaciones, pueden tomar decisiones y adquirir experiencia directa delante de unas situaciones que frecuentemente resultarían difícilmente accesibles a la realidad (control de una central nuclear, contracción del tiempo, pilotaje de un avión). También se pueden considerar simulaciones ciertos videojuegos que, al margen de otras consideraciones sobre los valores que incorporan (generalmente no muy positivos) facilitan el desarrollo de los reflejos, la percepción visual y la coordinación psicomotriz en general, además de estimular la capacidad de interpretación y de reacción ante un medio concreto.

En cualquier caso, **posibilitan un aprendizaje significativo por descubrimiento** y la investigación de los estudiantes/experimentadores puede realizarse en tiempo real o en tiempo acelerado, según el simulador, mediante preguntas del tipo: **¿Qué pasa al modelo si modifico el valor de la variable X? ¿Y si modifico el parámetro Y?** Se pueden diferenciar dos tipos de simuladores:

- *Modelos físico-matemáticos*, presentan de manera numérica o gráfica una realidad que tiene unas leyes representadas por un sistema de ecuaciones deterministas. Se incluyen aquí los programas-laboratorio, algunos trazadores de funciones y los programas que mediante un convertidor analógico-digital captan datos analógicos de un fenómeno externo al ordenador y presentan en pantalla un modelo del fenómeno estudiado o informaciones y gráficos que van asociados. Estos programas a veces son utilizados por profesores delante de la clase a manera de pizarra electrónica, como demostración o para ilustrar un concepto, facilitando así la transmisión de información a los alumnos, que después podrán repasar el tema interactuando con el programa.

- *Entornos sociales,* presentan una realidad regida por unas leyes no del todo deterministas. Se incluyen aquí los juegos de estrategia y de aventura, que exigen una destreza cambiante a lo largo del tiempo.

Constructores:

Son programas que tienen un entorno programable. Facilitan a los usuarios unos elementos simples con los cuales pueden construir elementos o entornos más complejos. De esta manera potencian el aprendizaje heurístico y, de acuerdo con las **teorías cognitivistas**, facilitan a los alumnos la construcción de sus propios aprendizajes, que surgirán a través de la reflexión que realizarán al diseñar programas y comprobar inmediatamente cuando los ejecuten, la relevancia de sus ideas. El proceso de creación que realiza el alumno genera preguntas del tipo: **¿Qué sucede si añado o elimino el elemento X?** Se pueden distinguir dos tipos de constructores:

- *Constructores específicos*, ponen a disposición de los estudiantes una serie de mecanismos de actuación (generalmente en forma de órdenes específicas) que les permiten llevar a cabo operaciones de un cierto grado de complejidad, mediante la construcción de determinados entornos, modelos o estructuras, y de esta manera avanzan en el conocimiento de una disciplina o entorno específico.
- *Lenguajes de programación,* como LOGO, PASCAL y BASIC, que ofrecen unos "laboratorios simbólicos" en los que se pueden construir un número ilimitado de entornos. Aquí los alumnos se convierten en profesores del ordenador. Además, con las interfaces convenientes, pueden controlar pequeños robots construidos con componentes convencionales (arquitecturas, motores), de manera que sus posibilidades educativas se ven ampliadas incluso en campos pre-tecnológicos. Así los alumnos pasan de un manejo abstracto de los conocimientos con el ordenador a una manipulación concreta y práctica en un entorno informatizado, que facilita la representación y comprensión del espacio y la previsión de los movimientos.

Dentro de este grupo de programas hay que destacar el lenguaje LOGO, creado en 1969 para Seymour Papert, que constituye el programa didáctico más utilizado en todo el mundo. LOGO es un programa constructor que tiene una doble dimensión:

- Proporciona **entornos de exploración** donde el alumno puede experimentar y comprobar las consecuencias de sus acciones, de manera que va construyendo un marco de referencia y esquemas de conocimiento, que facilitarán la posterior adquisición de nuevos conocimientos.
- Facilita una actividad formal y compleja, próxima al terreno de la construcción de estrategias de resolución de problemas: **la programación**. A través de ella los alumnos pueden establecer proyectos, tomar decisiones y evaluar los resultados de sus acciones.

Programas Herramientas:

Son programas que proporcionan un entorno instrumental con el cual se facilita **la realización de ciertos trabajos generales** de tratamiento de la información: escribir, organizar, calcular, dibujar, transmitir y captar datos. A parte de los lenguajes de autor (que también se podrían incluir en el grupo de los programas constructores), los más utilizados son programas de uso general que provienen del mundo laboral y, por tanto, quedan fuera de la definición que se ha dado de software educativo. No obstante, se han elaborado algunas versiones de estos programas "para niños" que limitan sus posibilidades a cambio de una, no siempre clara, mayor facilidad de uso. De hecho, muchas de estas versiones resultan innecesarias, ya que el uso de estos programas cada vez resulta más sencillo y cuando los estudiantes necesitan utilizarlos o su uso les resulta funcional, aprenden a manejarlos sin dificultad. Los programas más utilizados de este grupo son:

- *Procesadores de textos**,*** son programas que, con la ayuda de una impresora, convierten el ordenador en una fabulosa máquina de escribir. En el ámbito educativo debe hacerse una introducción gradual que puede empezar a lo largo de la Enseñanza Primaria, y ha de permitir a los alumnos familiarizarse con el teclado y con el ordenador en general, y sustituir parcialmente la libreta de redacciones por un disco (donde almacenarán sus trabajos). Al escribir con los procesadores de textos los estudiantes pueden concentrarse en el contenido de las redacciones y el resto de los trabajos que tengan encomendados despreocupándose por la caligrafía. Además el corrector ortográfico, que suelen incorporar, les ayudará a revisar las posibles faltas de ortografía antes de entregar el trabajo.

Además de este empleo instrumental, los procesadores de textos permiten realizar múltiples actividades didácticas, por ejemplo:

- Ordenar párrafos, versos, estrofas.

- Insertar frases y completar textos.
- Separar dos poemas.

- *Gestores de bases de datos*, sirven para generar potentes sistemas de archivo ya que permiten almacenar información de manera organizada y posteriormente recuperarla y modificarla. Entre las muchas actividades con valor educativo que se pueden realizar están las siguientes:

  - Revisar una base de datos ya construida para buscar determinadas informaciones y recuperarlas.
  - Recoger información, estructurarla y construir una nueva base de datos.

- *Hojas de cálculo*, son programas que convierten el ordenador en una versátil y rápida calculadora programable, facilitando la realización de actividades que requieran efectuar muchos cálculos matemáticos. Entre las actividades didácticas que se pueden realizar con las hojas de cálculo están las siguientes:
  - Aplicar hojas de cálculo ya programadas a la resolución de problemas de diversas asignaturas, evitando así la realización de pesados cálculos y ahorrando un tiempo que se puede dedicar a analizar los resultados de los problemas.
  - Programar una nueva hoja de cálculo, lo que exigirá previamente adquirir un conocimiento preciso del modelo matemático que tiene que utilizar.

- *Editores gráficos*, se emplean desde un punto de vista instrumental para realizar dibujos, portadas para los trabajos, murales y anuncios. Además, constituyen un recurso idóneo para desarrollar parte del currículum de Educación Artística: dibujo, composición artística y uso del color entre otros.
- *Programas de comunicaciones*, son programas que permiten que ordenadores lejanos (si disponen de módem) se comuniquen entre sí, a través de las líneas telefónicas y puedan enviarse mensajes, gráficos y programas. Desde una perspectiva educativa estos sistemas abren un gran abanico de actividades posibles para los alumnos, por ejemplo:

  - Comunicarse con otros compañeros e intercambiarse informaciones.

- Acceder a bases de datos lejanas para buscar determinadas informaciones.

- *Programas de experimentación asistida*, a través de variados instrumentos y convertidores analógico-digitales, recogen datos sobre el comportamiento de las variables que inciden en determinados fenómenos. Posteriormente con estas informaciones se podrán construir tablas y elaborar representaciones gráficas que representen relaciones significativas entre las variables estudiadas.
- *Lenguajes y sistemas de autor*, son programas que facilitan la elaboración de programas tutoriales a los profesores que no disponen de grandes conocimientos informáticos. Utilizan unas pocas instrucciones básicas que se pueden aprender en pocas sesiones. Algunos incluso, permiten controlar vídeos y dan facilidades para crear gráficos y efectos musicales, de manera que pueden generar aplicaciones multimedia. Algunos de los más utilizados en entornos PC han sido: PILOT, PRIVATE TUTOR, TOP CLASS, LINK WAY y QUESTION MARK.

  Después de exponer esta serie de aspectos significativos sobre el software educativo, se puede enmarcar el entorno del Software Educativo a nivel internacional como se representa en la Figura 6 que se muestra a continuación:

Figura 6. Entorno de trabajo del software educativo a nivel internacional

2.7 Características del Software Educativo en COPEXTEL

Teniendo en cuenta las características del proceso de enseñanza-aprendizaje cubano y el método histórico-lógico, se llega a la conclusión, que el software educativo producido por el COPEXTEL es mixto conteniendo elementos de:

- Tutoriales: Son programas que en mayor o menor medida dirigen, tutorizan el trabajo de los alumnos. En cualquier caso, son programas basados en los planteamientos conductistas de la enseñanza, que comparan las respuestas de los alumnos con los patrones que tienen como correctos, guían los aprendizajes de los estudiantes y facilitan la realización de prácticas más o menos rutinarias y su evaluación; en algunos casos una evaluación negativa genera una nueva serie de ejercicios de repaso. A partir de la estructura de su algoritmo, se distinguen cuatro categorías (Programas Lineales, Ramificado, Entornos Tutoriales y Sistema Tutoriales Expertos).

- Base de Datos: Proporcionan datos organizados en un entorno estático, según determinados criterios y facilitan su exploración y consulta selectiva. Pueden tener una estructura jerárquica, relacional o documental. En cualquier caso, según la forma de acceder a la información se pueden distinguir dos tipos de simulador (Base de Datos Convencionales y Base de Datos Tipo Experto).

- Constructor: Son programas con un entorno programable, que facilitan a los usuarios elementos simples, con los cuales pueden construir elementos o entornos más complejos. Así, potencian el aprendizaje heurístico y facilitan a los alumnos la construcción de sus propios aprendizajes, de acuerdo con las teorías cognitivistas. Se pueden distinguir dos tipos de constructores (Constructores Específicos y Lenguajes de Programación).

- Simulador: Presentan un modelo o entorno dinámico (generalmente a través de gráficos o animaciones interactivas), facilitan su exploración y modificación a los alumnos que pueden realizar aprendizajes inductivos o deductivos, mediante la observación y la manipulación de la estructura subyacente. Estos programas didácticos posibilitan un aprendizaje significativo por descubrimiento y la investigación de los estudiantes/experimentadores pueden realizarse en

tiempo real o en tiempo acelerado, según el simulador. Se pueden distinguir dos tipos de simulador (Modelo Físico-Matemático y Entorno Sociales).

Lo que hace que sea un software complejo..

2.8 Actores del modelo de desarrollo

El estudiante: eje central del proceso.

En cualquiera de los tipos de software a desarrollar, es esencial la participación activa del estudiante, bien sea como un elemento que permite modelar la acciones y procesos que debe desarrollar en un prototipo determinado, como actor autónomo del proceso de aprendizaje que determina no solo los tiempos y contendido de aprendizaje en un ambiente estructurado, sino también como evaluador y participante en la configuración de las actividades de aprendizaje y autor de los contenidos en el software de carácter evolutivo.

El papel del maestro.

Es claro que en esta mirada de desarrollo de software educativo, el maestro abandona su papel de usuario del mismo, para convertirse en un diseñador de ambientes de aprendizaje prototípicos o estructurados, y como participante activo y autor de software evolutivo. El docente ya no es exclusivamente un poseedor de información, su papel en el proceso educativo traspasa la transmisión de conocimiento y llega a los límites de la tutoría de los estudiantes, de la asesoría permanente, de la orientación de procesos y de un papel de par académico que tiene bajo su responsabilidad, e proceso de colaboración en el aprendizaje de los estudiantes y de estos entre si.

Los administradores de salas e ingenieros de soporte.

Esta mirada de desarrollo de software educativo, implica que los administradores de sala e ingenieros de soporte abandonan su postura de poder frente al manejo de programas, para convertirse en facilitadores de los procesos de desarrollo de ambientes propicios para el aprendizaje, que participan de forma colaborativa en los procesos de desarrollo de software poniendo sobre el tapete todos los

elementos técnicos, de manejo de programas y redes, necesarios para los procesos de aprendizaje de los estudiantes. En cambio de la búsqueda permanente de restricciones, se convierten en posibilitadores de procesos.

Directivos y administración Distrital.

Los procesos de desarrollo de software educativo deben responder a las necesidades del contexto escolar y los lineamientos determinados desde el PEI de las instituciones, por lo tanto el papel de los directivos y administradores Distritales es facilitar estos procesos de producción, ofreciendo todas las posibilidades para el desarrollo de las tres categorías de software descrito.

2.9 Descripción de la solución propuesta

Dada las disímiles características que pueden presentar un software educativo una propuesta de arquitectura no se puede limitar a escoger un patrón arquitectónico adecuado sino que debe ser además un algoritmo de trabajo para crear la arquitectura adecuada al proyecto en desarrollo

Pragmáticamente, al diseñar una arquitectura de software educativo se debe crear y representar componentes que interactúen entre ellos y tengan asignadas tareas específicas, además de organizarlos de forma tal que se logren los requerimientos establecidos. Esta arquitectura debe basarse en los patrones de soluciones ya probados, con la intención de no comenzar de cero las propuestas y utilizar modelos que han funcionado.

Es por ello que en la investigacion se decide proponer al MVC_E como el patrón arquitectónico de los Softaware Educativo en COPEXTEL (Figura 7).

La arquitectura Modelo-Vista-Controlador, es la implementación de la arquitectura en tres capas más extendida. Todas sus caracteristicas son extensible a la arquitectura en tres capas, con algunas pequeñas diferencias.

Definición del Patrón

Modelo (Model): Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada. El modelo debe de preservar la integridad de los datos.

Vista (View): Muestra la información al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador. Interactúa con la interfaz de usuario.

Controlador (Controller): Reciben las entradas, usualmente como eventos, e interpreta las operaciones del usuario; codificando los movimientos, pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio ("service requests") para el modelo o la vista. Es el que debe de controlar los eventos.

Responsabilidades de cada parte

Después de definir los conceptos de Modelo, Vista y Controlador, se definirán las responsabilidades de cada uno de ellos.

El modelo deberá:

Acceder a los datos persistentes, a la capa de almacenamiento de datos. Algo ideal es que este sea independiente de la Base de datos utilizada.

Llevar las estadísticas de sistema (en caso de haberlas).

Definir las reglas de negocio.

Notificar los cambios que se hayan hecho.

El controlador deberá:

Recibir los eventos de entrada.

Realizar la acción correspondiente al evento que se ha disparado.

Las vistas deberán:

Conseguir los datos que aporta el modelo y mostrárselos al usuario.

Saber cual es su controlador asociado, el cual puede pedirle algunos servicios típicos como el de actualizar.

Beneficios de utilizar el MVC_E:

• Menor acoplamiento.

- Desacopla las vistas de los modelos.
- Desacopla los modelos de la forma en que se muestran e ingresan los datos.

• Mayor cohesión.

- Cada elemento del patrón esta altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio).

• Las vistas proveen mayor flexibilidad y agilidad.

- Se puede crear múltiples vistas de un modelo.
- Las vistas pueden anidarse.
- Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual.
- Se puede sincronizar las vistas.
- Las vistas pueden concentrarse en diferentes aspectos del modelo.

• Mayor facilidad para el desarrollo de clientes ricos en multiples dispositivos y canales

- Una vista para cada dispositivo que puede variar según sus capacidades.
- Una vista para la Web y otra para aplicaciones de escritorio.

• Más claridad de diseño.

• Facilita el mantenimiento.

• Mayor escalabilidad.

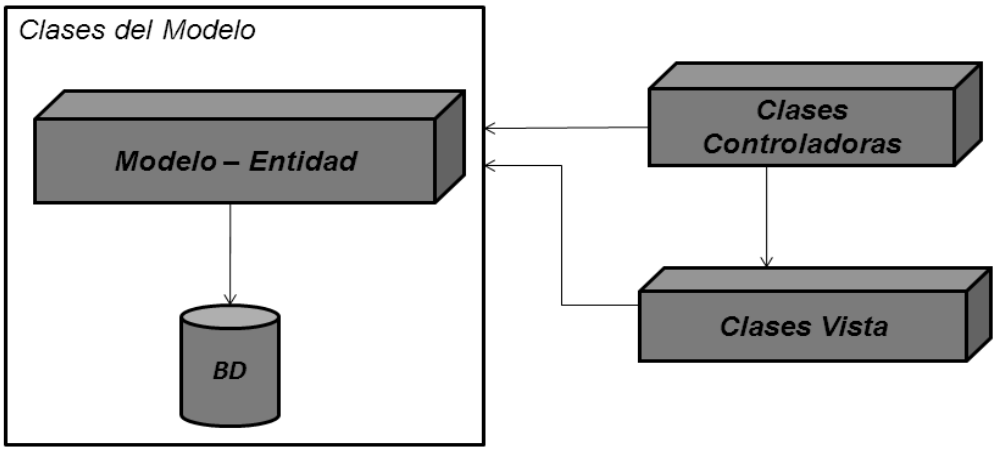

Figura 7. El MVC_E como patrón arquitectónicos del software educativo COPEXTEL

2.10 Conclusiones

En este capítulo se desarrolló la propuesta de solución, obteniéndose un patrón de arquitectura despues de un análisis profundo para un mejor desarrollo de la realizacion de software educativo. Al mismo tiempo se ha podido arribar a las siguientes conclusiones:

- El software educativo cubano, tiene marcadas diferencias en comparación con los desarrollados en otros países del mundo.
- El software educativo que se produce en COPEXTEL tiene una estructura general común. Sus componentes se agrupan en tres módulos principales:

  El que gestiona la comunicación con el usuario.

  El que contiene debidamente organizados los contenidos informáticos del programa (BD).

  El que tramita las actuaciones del ordenador y sus respuestas a las acciones de los usuarios.
- Se han desarrollado a lo largo de los años, desde la presentación del MVC a la comunidad científica variantes fundamentales, como el MVC_E, que en la actualidad logra un mayor acercamiento al software cubano.
- Se propone al MVC_E como patrón arquitectónico para modelar el software educativo que se produce en COPEXTEL.

Capítulo 3: "Validación de la Propuesta de Solución"

3.1 Introducción

Para realizar la validación de la propuesta de solución, se utilizó el Método de Experto el cuál se basa en la evaluación cuantitativa de criterios definidos, que permite realizar un estudio de expertos para determinar si se acepta o no la propuesta analizada.

3.2 Guía para la Validación

Para realizar la validación de la propuesta de solución se llevaron a cabo un conjunto de pasos, los cuales se describen a continuación:

1. Se elaboraron los criterios de evaluación de acuerdo a las características de la propuesta y se organizaron por grupos.

Grupo No. 1: Criterios de mérito científico

1. Valor científico de la propuesta.
2. Calidad de la investigación.
3. Contribución científica.
4. Responsabilidad científica y profesionalidad del investigador.

Grupo No. 2: Criterios de implantación

5. Satisfacción de las necesidades de los ingenieros de software.
6. Necesidad del empleo de la propuesta.

Grupo No.3: Criterios de flexibilidad

7. Posibilidades de aplicación.
8. Impacto en el área a la cuál está destinada.

Grupo No.4: Criterios de impacto

9. Repercusión en los proyectos productivos.

10. Organización en el proceso de documentación.

2. Se le asignó un peso relativo a cada grupo de criterios de acuerdo al porciento que representa cada grupo del total y los intereses a evaluar.

Grupo No. 1…….. 40
Grupo No. 2…….. 20
Grupo No.3……. 20
Grupo No.4……...20

3. Se organiza un comité de expertos con una cantidad mínima de 7 teniendo en cuenta su especialidad, grado científico y currículo.

4. Se les entrega a los expertos un resumen de la propuesta para que estudien el tema a evaluar y dos modelos, uno para que valore el peso relativo de cada criterio (Anexo1) y otro para realizar una evaluación cuantitativa de cada criterio con una escala de 1-5 (Anexo 2) y la apreciación cualitativa con una clasificación final del proyecto en excelente, bueno, aceptable, cuestionable y malo. También se da la posibilidad de dar su opinión haciendo una valoración final del proyecto, emitiendo todas aquellas consideraciones que estimaron convenientes.

5. Después de recibir los valores del peso relativo de cada criterio se construye la Tabla 2.

Tabla 2. Resultado del trabajo de expertos

G	C/E	E1	E2	E3	E4	E5	E6	E7	Ep
40	C1	8	10	10	5	8	10	10	8.71
	C2	15	10	10	15	11	9	11	11.6
	C3	8	10	10	5	10	10	8	8.71

	C4	9	10	10	15	11	11	11	11
20	**C5**	15	10	10	10	10	11	10	10.8
	C6	5	10	10	10	10	9	10	9.14
20	**C7**	10	15	15	15	11	10	11	12.4
	C8	10	5	5	5	9	10	9	7.57
20	**C9**	10	10	10	10	10	10	10	10
	C10	10	10	10	10	10	10	10	10
T		100	100	100	100	100	100	100	100

6. Se verifica la consistencia en el trabajo de los expertos, para lo que se utiliza el coeficiente de concordancia de Kendall y el estadígrafo Chi cuadrado (X^2). Se sigue el procedimiento siguiente:

- Sea C el número de criterios que van a evaluarse y E el número de expertos que realizan la evaluación.
- Para cada criterio se determina:

  ∑E: Sumatoria del peso dado por cada experto

  Ep: Puntuación promedio del peso dado por cada experto

  M∑E: media de los ∑E

  ΔC: Diferencia entre ∑E y M∑E
- Se determina la desviación de la media, que posteriormente se eleva al cuadrado para obtener la dispersión (S) por la expresión

  $S = \sum (\sum E - \sum \sum E / C)^2$

Tabla 3. Cálculo de la Dispersión (S) para hallar la concordancia entre los expertos

	∑E	**∑E / C**	**∑E - ∑ ∑E / C**	**(∑E - ∑ ∑E / C)²**

C1	61	6.1	-9	81
C2	81	8.1	11	121
C3	61	6.1	-9	81
C4	77	7.7	7	49
C5	76	7.6	6	36
C6	64	6.4	-6	36
C7	87	8.7	17	289
C8	53	5.3	-17	289
C9	70	7	0	0
C10	70	7	0	0
∑ ∑E / C	-	70	-	-
S = ∑ (∑E - ∑ ∑E / C)^2	-	-	-	982

- Conociendo la dispersión se puede calcular el coeficiente de concordancia de Kendall (W)

  $W = S / E^2 (C^3 – C)/ 12$

- El coeficiente de concordancia de Kendall permite calcular el Chi cuadrado real

  $X^2 = E (C-1) W$

- Los valores obtenidos se muestran en la Tabla 4.

Tabla 4. Tabla para el cálculo de concordancia de Kendall

Expertos/Criterios	**E_1**	**E_2**	**E_3**	**E_4**	**E_5**	**E_6**	**E_7**	**∑E**	**Ep**	**ΔC**	**ΔC^2**
C_1	8	10	10	5	8	10	10	61	8.71	9	81
C_2	15	10	10	15	11	9	11	81	11.6	11	121

C_3	8	10	10	5	10	10	8	61	8.71	9	81
C_4	9	10	10	15	11	11	11	77	11	7	49
C_5	15	10	10	10	10	11	10	76	10.8	6	36
C_6	5	10	10	10	10	9	10	64	9.14	6	36
C_7	10	15	15	15	11	10	11	87	12.4	17	289
C_8	10	5	5	5	9	10	9	53	7.57	17	289
C_9	10	10	10	10	10	10	10	70	10	0	0
C_{10}	10	10	10	10	10	10	10	70	10	0	0
DC	100	100	100	100	100	100	100	700	99.93	82	982
M ∑E	70										
W	0.24										
X^2	15.12										
$X^2_{(\alpha, c-1)}$	21.6660										

- El Chi cuadrado calculado se compara con el obtenido del las tablas estadísticas
- Si se cumple:

  $X^2_{real} < X^2_{(\alpha, c-1)}$

  $15.12 < 21.6660$

Existe concordancia en el trabajo de expertos

7. Después de comprobar la consistencia del trabajo de expertos se puede definir el peso relativo de cada criterio (P).

8. Conociendo el peso de cada criterio y la calificación dada por los evaluadores en una escala de 1-5 se puede construir la Tabla 5, para obtener el valor de de P × c., donde (c), es el criterio promedio concebido por los expertos.

Tabla 5. Tabla de calificación de cada criterio

Criterios	**Calificación (c)**					**P**	**P × c**
	1	**2**	**3**	**4**	**5**		
C_1				X		0.0871	0.3484
C_2				X		0.116	0.464
C_3				X		0.0871	0.3484
C_4					X	0.11	0.55
C_5					X	0.108	0.54
C_6					X	0.0914	0.457
C_7					X	0.124	0.62
C_8				X		0.0757	0.3008
C_9					X	0.1	0.5
C_{10}				X		0.1	0.4
Total							4.5286

9. Se calcula el Índice de aceptación del proyecto (IA).

$$IA = \sum (P \times c) / 5$$

$$IA = 0.905$$

10. Por último se determina la probabilidad de éxito de la propuesta

El Índice de Aceptación calculado es 0.905.

Rangos predefinidos de Índice de Aceptación.

IA > 0,7 Existe alta probabilidad de éxito

0,7 > IA > 0,5 Existe probabilidad media de éxito

0,5 > IA > 0,3 Probabilidad de éxito baja

0,3 > IA Fracaso seguro

Por lo que la probabilidad de éxito es alta.

3.3 Conclusiones

En este capítulo se evaluó la guía obtenida mediante el Método de Expertos; el cual permitió analizar los criterios de cada uno de los expertos y determinar el índice de aceptación que tiene la propuesta del presente trabajo, obteniéndose concordancia en el trabajo de expertos y una alta probabilidad de éxito de ser aplicada en las entidades que se dedican a evaluar la calidad de los productos de software.

Conclusiones Generales

Con el desarrollo de este trabajo de investigación, se obtuvo una documento con las características, elementos estructurales y funcionales del software educativo y se propuso un patrón de arquitectura para mejorar la calidad y usabilidad del software educativo en la empresa Sys - COPEXTEL.

Finalmente se hizo la validación de la propuesta, arrojando que la probabilidad de éxito es alta, lo que implica desde el punto de vista teórico, el cumplimiento de la idea a defender planteada.

Recomendaciones

Los objetivos planteados en el presente trabajo se alcanzaron de manera general. Se considera que para seguir solucionando los problemas respecto a la utilización de arquitectura del software se necesita de un proceso investigativo mucho más amplio y este trabajo es sólo un primer nivel de madurez del mismo, teniendo en cuenta esto se pueden hacer las siguientes recomendaciones:

- Aplicar lo planteado en el presente trabajo en el proceso de desarrollo del software educativo que se produce en la empresa Sys - COPEXTEL, para probar su veracidad y efectividad**.**
- Extender la solución a otros proyectos productivos de la universidad, que desarrollen software educativo.
- Realizar talleres, seminarios, conferencias, etc., donde se deje clara la importancia de la aplicación de las arquitecturas en la realización de software educativos.

Referencias Bibliográficas

A, Rodríguez Liván. SOFTWARE EDUCATIVO. HACIA UNA NUEVA PEDAGOGÍA BASADA EN LAS TIC. [Online] [Cited: 4 25, 2008.] http://rvarela.ispvc.rimed.cu/articulos/rv1801.pdf.

Barrera Yanes, Rafael. 1998. *Del objetivo al guión interactivo.* La Habana : s.n., 1998. Vol. Vol 1.

Cataldi, Zulma. 2000. Metodología de diseño, desarrollo y evaluación de software educativo. [Online] 2000.Tesis de Magíster en Informática. (versión resumida).Facultad de Informática. http://www.fi.uba.ar/laboratorios/lsi/cataldi-tesisdemagistereninformatica.pdf.

Fajardo, Elena Galán. El guión didáctico para materiales multimedia. [Online] [Cited: 2 4, 2008.] Universidad Carlos III de Madrid. http://www.ucm.es/info/especulo/numero34/guionmu.html.

Gregorio R.G., Javier G.F.and Eduardo G.J. 2008. metodologia de investigacion cualitativa. [Online] 2 6, 2008. http://www.lapaginadelprofe.cl/guiatesis/31icualitativa.htm.

Huertas, Pérez. 1998. Introducción a la multimedia: realización y producción de programas. [Online] 1998. [Cited: 3 4, 2008.] Madrid: IORTV.

John C. Georgas, Eric M. Dashofy, y Richard N. Taylor. Desarrollo Centrado en la Arquitectura: Un acercamiento diferente a la Ingeniería de Software. [Online] [Cited: 2 4, 2008.] http://www.acm.org/crossroads/espanol/xrds12-4/arqcentric.html.

Kruchten, Philippe. 1995. *The 4+1 View Model of Architecture.* 1995. Vol. vol. 12.

2008. La Historia del Software. [Online] 3 27, 2008. http://cellular.ci.ulsa.mx/comun/historiaw/node23.html.

Lara, Luis R. Introducción a un modelo complejo de los softwares multimediales educativos. [Online] [Cited: 3 5, 2008.] Facultad de Ciencias Exactas y Naturales de la Universidad Nacional de Catamarca.Proyecto de Investigación PROTEO. http://www.um.es/ead/red/12/lara.pdf .

Manuel Gértrudix Barrio, Sergio Álvarez García,Antonio Galisteo del Valle,María del Carmen Gálvez de la Cuesta,Felipe Gértrudix Barrio. 2007. Acciones de diseño y desarrollo de objetos

educativos digitales:programas institucionales. [Online] abril 2007. artículo. http://www.uoc.edu/rusc/4/1/dt/esp/gertrudix_alvarez_galisteo_galvez.pdf.

Marqués,Pere.1996. El software educativo.[Online]1996. http://www.lmi.ub.es/te/any96/marques_software/.

MORA,A.J.H. Multimedia.[Online][Cited:415,2008.] http://www.monografias.com/trabajos7/mult/mult2.shtml#his.

Nurileidis Almeida Cintra and Yoannia Viera Cisneros. 2007. *Principios para la evaluación y certificación de la calidad de los productos de Software Educativo en la Universidad de las Ciencias Informáticas.* Ciudad de la Habana : s.n., 2007. TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

Pérez Huertas, F. J. 1998. *Introducción a la multimedia: realización y producción de programas.* 1998. (Unidad didáctica 158).

Pressman, Roger S. 2002. *Ingeniería de Software. Un enfoque práctico (Traducción del original en inglés: Software Engeering A practical Aproach).* 5ta edición. Mac Graw Hill : s.n., 2002.

Ricardo, Febe Angel Ciudad. 2006. II TALLER DE SOFTWARE EDUCATIVO E HIPERMEDIA. [Online] Abril 2006. Ciudad de la Habana,Cuba. http://www.informaticahabana.com/evento_virtual/files/MUL053.pdf.

2006. PRESENTACIÓN DEL LENGUAJE DE MODELACIÓN PARA APLICACIONES EDUCATIVAS ApEM – L 1.0. [Online] 2006. Ciudad de La Habana. http://virtualst.fordes.co.cu/EVirtual/files/Trabajo_23.doc.

Utilización del Patrón Modelo – Vista – Controlador (MVC) en el diseño de software educativos. [Online] [Cited: 5 4, 2008.] http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista.shtml.

Río, Agustín Cernuda. 2002. Sistema de verificación de componentes software. [Online] Febrero 2002. TESIS DOCTORAL.Departamento de Informática.UNIVERSIDAD DE OVIEDO. http://willydev.net/.../2008.05.07.Tesis.sistema%20de%20verificacion%20de%20componentes%20software-tesis.pdf.

2008. Software. [Online] 4 21, 2008. http://es.wikipedia.org/wiki/Software.

Standish. 1994. The CHAOS Report. [Online] 1994. http://www.standishgroup.com/sample_research/chaos.

SUÁREZ, D. A. D. V. M. ANTECEDENTES PEDAGÓGICOS DEL USO DE LA TECNOLOGÍA MULTIMEDIA EN LA EDUCACIÓN. [Online] [Cited: 5 20, 2008.] http://www.servicio.cid.uc.edu.ve/educacion/revista/a5n26/5-26-10.pdf.

UNESCO. 2001. Ediciones UNESCO. Santillana : s.n., 2001.

VAQUERO, A. 1997. *«Las TIC para la enseñanza, la formación y el aprendizaje».* 1997. En Revista Novatita 132: Monografía sobre las TIC en la Educación.

ZULMA CATALDI, FERNANDO LAGE,RAÚL PESSACQ,RAMÓN GARCÍA–MARTÍNEZ. METODOLOGÍA EXTENDIDA PARA LA CREACIÓN DE SOFTWARE EDUCATIVO DESDE UNA VISIÓN INTEGRADORA. [Online] [Cited: 5 6, 2008.] Buenos Aires, Argentina.Volumen 2. REVISTA LATINOAMERICANA DE TECNOLOGÍA EDUCATIVA. http://www.itba.edu.ar/capis/webcapis/RGMITBA/articulosrgm/R-extremadura-2.pdf.

Bibliografía

A, Rodríguez Liván. SOFTWARE EDUCATIVO. HACIA UNA NUEVA PEDAGOGÍA BASADA EN LAS TIC. [Online] [Cited: 4 25, 2008.] http://rvarela.ispvc.rimed.cu/articulos/rv1801.pdf.

Barrera Yanes, Rafael. 1998. Del objetivo al guión interactivo. La Habana : s.n., 1998. Vol. Vol 1. Cataldi, Zulma. 2000. Metodología de diseño, desarrollo y evaluación de software educativo. [Online] 2000. Tesis de Magíster en Informática. (versión resumida).Facultad de Informática. http://www.fi.uba.ar/laboratorios/lsi/cataldi-tesisdemagistereninformatica.pdf.

Fajardo, Elena Galán. El guión didáctico para materiales multimedia. [Online] [Cited: 2 4, 2008.] Universidad Carlos III de Madrid. http://www.ucm.es/info/especulo/numero34/guionmu.html.

Gregorio R.G., Javier G.F.and Eduardo G.J. 2008. metodologia de investigacion cualitativa. [Online] 2 6, 2008. http://www.lapaginadelprofe.cl/guiatesis/31icualitativa.htm.

Huertas, Pérez. 1998. Introducción a la multimedia: realización y producción de programas. [Online] 1998. [Cited: 3 4, 2008.] Madrid: IORTV.

John C. Georgas, Eric M. Dashofy, y Richard N. Taylor. Desarrollo Centrado en la Arquitectura: Un acercamiento diferente a la Ingeniería de Software. [Online] [Cited: 2 4, 2008.] http://www.acm.org/crossroads/espanol/xrds12-4/arqcentric.html.

Kruchten, Philippe. 1995. The 4+1 View Model of Architecture. 1995. Vol. vol. 12. 2008. La Historia del Software. [Online] 3 27, 2008. http://cellular.ci.ulsa.mx/comun/historiaw/node23.html.

Lara, Luis R. Introducción a un modelo complejo de los softwares multimediales educativos. [Online] [Cited: 3 5, 2008.] Facultad de Ciencias Exactas y Naturales de la Universidad Nacional de Catamarca.Proyecto de Investigación PROTEO. http://www.um.es/ead/red/12/lara.pdf .

Manuel Gértrudix Barrio, Sergio Álvarez García,Antonio Galisteo del Valle,María del Carmen Gálvez de la Cuesta,Felipe Gértrudix Barrio. 2007. Acciones de diseño y desarrollo de objetos educativos digitales:programas institucionales. [Online] abril 2007. artículo. http://www.uoc.edu/rusc/4/1/dt/esp/gertrudix_alvarez_galisteo_galvez.pdf.

Marqués, Pere. 1996. El software educativo. [Online] 1996. http://www.lmi.ub.es/te/any96/marques_software/.

MORA,A.J.H. Multimedia. [Online] [Cited: 4 15, 2008.] http://www.monografias.com/trabajos7/mult/mult2.shtml#his.

Nurileidis Almeida Cintra and Yoannia Viera Cisneros. 2007. Principios para la evaluación y certificación de la calidad de los productos de Software Educativo en la Universidad de las Ciencias Informáticas. Ciudad de la Habana : s.n., 2007. TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

Pérez Huertas, F. J. 1998. Introducción a la multimedia: realización y producción de programas. 1998. (Unidad didáctica 158).

Pressman, Roger S. 2002. Ingeniería de Software. Un enfoque práctico (Traducción del original en inglés: Software Engeering A practical Aproach). 5ta edición. Mac Graw Hill : s.n., 2002.

Ricardo, Febe Angel Ciudad. 2006. II TALLER DE SOFTWARE EDUCATIVO E HIPERMEDIA. [Online] Abril 2006. Ciudad de la Habana,Cuba. http://www.informaticahabana.com/evento_virtual/files/MUL053.pdf.

2006. PRESENTACIÓN DEL LENGUAJE DE MODELACIÓN PARA APLICACIONES EDUCATIVAS ApEM – L 1.0. [Online] 2006. Ciudad de La Habana. http://virtualst.fordes.co.cu/EVirtual/files/Trabajo_23.doc.

Utilización del Patrón Modelo – Vista – Controlador (MVC) en el diseño de software educativos. [Online] [Cited: 5 4, 2008.]

http://www.monografias.com/trabajos43/patron-modelo-vista/patron-modelo-vista.shtml

Río, Agustín Cernuda. 2002. Sistema de verificación de componentes software. [Online] Febrero 2002. TESIS DOCTORAL.Departamento de Informática.UNIVERSIDAD DE OVIEDO. http://willydev.net/.../2008.05.07.Tesis.sistema%20de%20verificacion%20de%20componentes%20software-tesis.pdf

2008. Software. [Online] 4 21, 2008. http://es.wikipedia.org/wiki/Software.
Standish. 1994. The CHAOS Report. [Online] 1994.
http://www.standishgroup.com/sample_research/chaos.

SUÁREZ, D. A. D. V. M. ANTECEDENTES PEDAGÓGICOS DEL USO DE LA TECNOLOGÍA MULTIMEDIA EN LA EDUCACIÓN. [Online] [Cited: 5 20, 2008.] http://www.servicio.cid.uc.edu.ve/educacion/revista/a5n26/5-26-10.pdf

UNESCO. 2001. Ediciones UNESCO. Santillana : s.n., 2001.

V.C., Nurileidis A.C.AND Yoannia. 207. Principios para la evaluación y certificación de la calidad de los productos de Software Educativo en la Universidad de las Ciencias Informáticas. Ciudad de La Habana : s.n., 207. TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS.

VAQUERO, A. 1997. «Las TIC para la enseñanza, la formación y el aprendizaje». 1997. En Revista Novatita 132: Monografía sobre las TIC en la Educación.

ZULMA CATALDI, FERNANDO LAGE,RAÚL PESSACQ,RAMÓN GARCÍA–MARTÍNEZ. METODOLOGÍA EXTENDIDA PARA LA CREACIÓN DE SOFTWARE EDUCATIVO DESDE UNA VISIÓN INTEGRADORA. [Online] [Cited: 5 6, 2008.] Buenos Aires, Argentina.Volumen 2. REVISTA LATINOAMERICANA DE TECNOLOGÍA EDUCATIVA. http://www.itba.edu.ar/capis/webcapis/RGMITBA/articulosrgm/R-extremadura-2.pdf

Glosario de Términos

Aprendizaje: El aprendizaje es uno de los procesos más importantes para la psicología científica actual, es un cambio casi permanente en el comportamiento del organismo, mediante el cual es posible modificar lo que se ha aprendido anteriormente**.** A diferencia de los animales, que nacen con instrucciones genéticas para la supervivencia, los humanos tenemos la capacidad de aprendizaje, la cual nos da más flexibilidad para adaptarnos al medio ambiente. Podemos aprender a resguardarnos de cambios climáticos y adaptarnos a cualquier ambiente, nuestra capacidad de aprendizaje nos permite afrontar cambios.

Cognitivo: Proceso exclusivamente intelectual que precede al aprendizaje, las capacidades cognitivas solo se aprecian en la acción, es decir primero se procesa información y después se analiza, se argumenta, se comprende y se producen nuevos enfoques. El desarrollo de lo cognitivo en el alumno debe ser el centro del proceso de enseñanza por parte del docente. Este término es utilizado por la psicología moderna, concediendo mayor importancia a los aspectos intelectuales que a los afectivos y emocionales, en este sentido se tiene un doble significado: primero, se refiere a una representación conceptual de los objetos y segundo, es la comprensión o explicación de los objetos.

Hipertexto: Un hipertexto es un documento digital o no, que se puede leer de manera no secuencial. Un hipertexto tiene los siguientes elementos: secciones, enlaces o hipervínculos y anclajes.

Aprendizaje significativo: El aprendizaje significativo es un aprendizaje relacional. El sentido lo da la relación del nuevo conocimiento con: conocimientos anteriores, situaciones cotidianas, la propia experiencia y situaciones reales.

Ingeniería del software: Aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software.

Módulo: Un módulo es un componente auto controlado de un sistema, el cual posee una interfaz bien definida hacia otros componentes.

Peso: nivel de importancia asignado a cada una de las actividades que se pretenden medir.

Proceso: Conjunto de actividades, realizadas en forma secuencial, que realiza una organización, para crear, producir y entregar productos, de tal manera que satisfagan las necesidades de sus clientes.

Producto: Resultado concreto, observable y medible que surge como consecuencia del proceso, proyecto o experiencia desarrollada.

Proyecto: Es un elemento organizativo a través del cual se gestiona el desarrollo de software, su resultado es un producto.

Software Educativo: Se define como un programa automatizado diseñado con una intencionalidad educativa para ser utilizado en el proceso de aprendizaje.

SIGLAS

TIC: Tecnologías de la Información y las Comunicaciones.

SWE: Software Educativo.

UCI: Universidad de las Ciencias Informáticas.

POSA: Patrones de Arquitectura Orientada a Servicios.

UML: Lenguaje Unificado de Modelado.

CD: Disco Compacto.

CES: Consumer Electronic Show.

CSP: Communicating Sequential Process.

CRC: Clase_Responsabilidad_Contrato.

Anexos

Anexo 1

Modelo No. 1

Guía para informar el peso de los criterios.

Fecha de recepción _____

Fecha de entrega _____

Nombre y Apellidos del evaluador _____

Le otorgará un peso a cada criterio de acuerdo a su opinión y el peso total de cada grupo debe sumar:

Grupo No.1……. 40

Grupo No.2……. 20

Grupo no.3……. 20

Grupo No.4…….20

Para que el peso total asignado sea 100.

Grupo No. 1: Criterios de mérito científico

1. Valor científico de la propuesta.

Peso…….

2. Calidad de la investigación.

Peso…….

3. Contribución científica.

Peso…….

4. Responsabilidad científica y profesionalidad del investigador.

Peso…….

Grupo No. 2: Criterios de implantación

5. Satisfacción de las necesidades de los ingenieros de software.

Peso…….

6. Necesidad del empleo de la propuesta.

Peso…….

Grupo No.3: Criterios de flexibilidad

7. Posibilidades de aplicación.

Peso…….

8. Impacto en el área a la cuál está destinada.

Peso…….

Grupo No.4: Criterios de impacto

9. Repercusión en los proyectos productivos.

Peso…….

10. Organización en el proceso de documentación.

Peso…….

Anexo 2

Modelo No. 2

Guía para la evaluación.

Fecha de recepción _____

Fecha de entrega _____

Nombre y Apellidos del evaluador_____

• Criterios de medida que se evalúan en una escala de 1 – 5

Grupo No. 1: Criterios de mérito científico

1. Valor científico de la propuesta.

Evaluación…….

2. Calidad de la investigación.

Evaluación …….

3. Contribución científica.

Evaluación …….

4. Responsabilidad científica y profesionalidad del investigador

Evaluación …….

Grupo No. 2: Criterios de implantación

5. Satisfacción de las necesidades de los ingenieros de software.

Evaluación …….

6. Necesidad del empleo de la propuesta.

Evaluación …….

Grupo No.3: Criterios de flexibilidad

7. Posibilidades de aplicación.

Evaluación …….

8. Impacto en el área a la cuál está destinada.

Evaluación …….

Grupo No.4: Criterios de impacto

9. Repercusión en los proyectos productivos.

Evaluación …….

10. Organización en el proceso de documentación.

Evaluación …….

• Categoría final de la propuesta.

___ Excelente: Alta novedad científica, con aplicabilidad y resultados relevantes.

___ Bueno: Novedad científica, resultados destacados.

___ Aceptable: Suficientemente bueno con reservas.

___ Cuestionable: No tiene relevancia científica y los resultados son malos.

___ Malo: No aplicable.

• Valoración final

Sugerencias del evaluador para mejorar la calidad del proyecto

Elementos críticos que deben mejorarse.