

---

## MeISE: Metodología de Ingeniería de Software Educativo

M.C. Ma. Antonieta Abud Figueroa  
aabud@prodigy.net.mx

*Instituto Tecnológico de Orizaba  
Orizaba, Veracruz, México*

---

### Resumen

Es conocido que la aplicación de una metodología adecuada al tipo de software a desarrollar facilita su elaboración y conlleva a la obtención de un producto de calidad. En este artículo se propone una metodología para la construcción de aplicaciones de software educativo, incorporando las mejores prácticas de la ingeniería de software y del diseño instruccional. Se busca ofrecer una guía metodológica que asegure un producto de software educativo de calidad que cumpla con las características de funcionalidad, usabilidad y fiabilidad, características deseables y necesarias para un material educativo interactivo.

### Palabras clave

Software educativo, metodología, ingeniería de software

---

### Introducción

En los últimos años, el avance en las tecnologías de información y las comunicaciones tiene influencia en la transmisión del conocimiento. El desarrollo de software capaz de ayudar al estudiante a adquirir y afianzar sus conocimientos en diversas áreas impulsa la investigación en el área de software educativo, tanto en la parte metodológica como en la parte tecnológica.

Uno de los principales problemas en la construcción del software educativo es seguir un proceso de desarrollo que asegure su calidad. Se requiere incluir en su diseño criterios que favorezcan la comprensión del contenido por parte del alumno; es decir, deben apoyarse en bases psicopedagógicas sobre el aprendizaje (conductista, cognitivista, constructivista) así como en los principios básicos de la ingeniería de software que permitan concretar el desarrollo de la aplicación en forma exitosa. Las metodologías convencionales de ingeniería de software generalmente abarcan actividades para la obtención de los requisitos, el diseño del sistema (diseño preliminar y diseño detallado), la construcción, las pruebas, la instalación y el mantenimiento del producto de software, todas enfocadas a atender los aspectos técnicos del producto y no se ocupan de los aspectos de la calidad didáctica, por lo que es necesario adaptarlos para que incluyan actividades orientadas a atender las características didácticas.

Actualmente existen propuestas de metodologías para la elaboración de software educativo como las de (Galvis, 2000), (Hinojosa, 1998), (Peláez y López, 2006) y (Cataldi, 2006) que guían su proceso de diseño, desarrollo y evaluación; sin embargo la mayoría se centran en la parte del diseño pedagógico y desatienden los aspectos computacionales, aunque

algunas, como las propuestas de (Gómez et al. 1998) y (Díaz-Antón et al., 2002) incluyen un enfoque global.

En este artículo se presenta la propuesta de una metodología de desarrollo de software educativo que propone un enfoque iterativo e incluye aspectos computacionales, pedagógicos y de comunicación. Se describen las etapas del ciclo de vida y las actividades y artefactos a obtener en el desarrollo del producto. Se describe además, un prototipo para apoyo al aprendizaje de factorización en álgebra elaborado bajo esta metodología.

### Metodología de Ingeniería de Software Educativo (MeISE)

La Metodología de Ingeniería de Software Educativo MeISE propone un ciclo de vida dividido en dos etapas. En la primera etapa se contempla la definición de requisitos y el análisis y diseño preliminar, durante los cuales se determinan en forma global las características que se pretende alcanzar con el producto, los requisitos pedagógicos, de comunicación y la arquitectura sobre la cual se construirá el software, y se termina con un plan de iteraciones las cuales se programan teniendo cuidado de que el producto que se libera al término de cada una está didácticamente completo, es decir que cubre completamente algunos de los objetivos didácticos del software. Una vez establecidos estos lineamientos, inicia la segunda etapa, en la cual se procede a desarrollar el producto, de modo que el equipo toma cada iteración, la diseña, la construye, la prueba y la implementa, evaluando al final la conveniencia de proseguir con subsecuentes iteraciones hasta obtener un producto completo.

Las fases propuestas para la etapa de definición son: la fase conceptual, durante la cual se identifican los requerimiento del sistema, se conforma el equipo de trabajo y se elabora el plan de desarrollo; la fase de análisis y diseño inicial, en la que se propone la arquitectura que servirá de base para la solución del problema y se establecen las características pedagógicas y de comunicación que regirán el desarrollo del software; finalmente la fase de plan de iteraciones, en la cual se divide el proyecto en partes funcionales que permitan mejor control en su desarrollo. En la etapa de desarrollo se tienen: la fase de diseño computacional, en la que se realizará un diseño computacional detallado de un incremento específico del software; la fase de desarrollo, durante la cual se implementa la arquitectura en forma incremental (iteración por iteración); y la fase de despliegue, donde se realiza la transición del producto ejecutable al usuario final. Estas tres últimas etapas se repiten iterativamente para cada incremento del software. El modelo se ilustra en la figura 1.

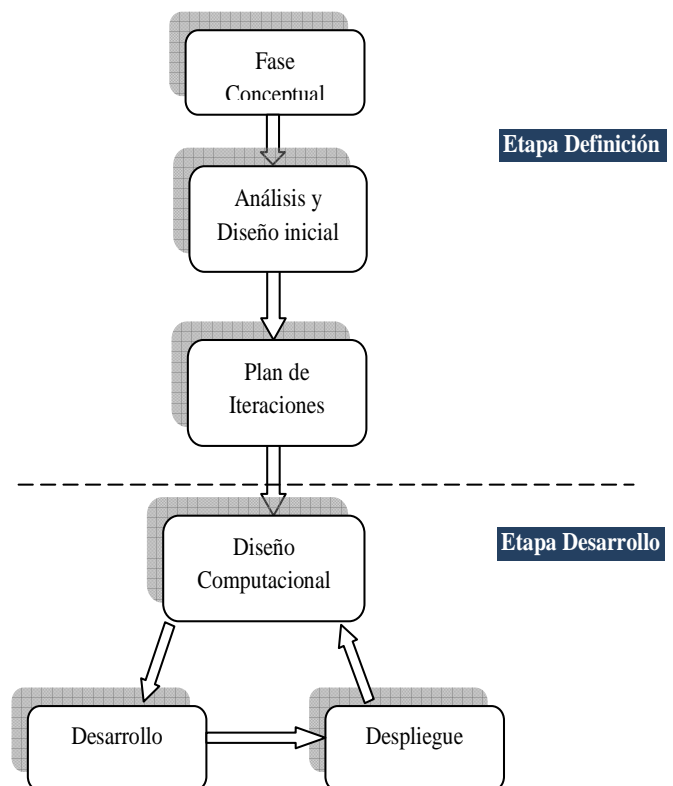


Figura 1. Ciclo de vida de la Metodología

## Descripción de Fases

### *Etapa conceptual*

Esta etapa inicia con una investigación sobre los requerimientos que se cubrirán con el producto a desarrollar, delimitando su alcance. Se desarrolla el plan del proyecto, se evalúan riesgos y se establecen los criterios de éxito. En la tabla 1 se muestran las actividades a realizar y los artefactos que se generan en esta fase.

*Tabla 1. Actividades y Artefactos de la Fase Conceptual.*

<b>ACTIVIDAD</b>	<b>ARTEFACTO</b>
Analizar las necesidades educativas	<u>Modelo instruccional</u> (incluye temática a atender, objetivos, conocimientos previos, fuentes de información, modelo educativo a utilizar, elementos de motivación y formas de evaluación) <u>Glosario</u> (descripción de los términos que pueden causar confusión o duda)
Revisar alternativas de solución	<u>Estudio de alternativas</u> (establece las diferentes alternativas que se tienen para el desarrollo del software, se determina el tipo de modelo educativo y se justifica la elección)
Elaborar un estudio de riesgos	<u>Lista de riesgos</u> (establece los riesgos relativos al desarrollo y a los aspectos pedagógicos y la forma de atenderlos)
Conformar del equipo de trabajo y el plan inicial de desarrollo	<u>Plan Inicial</u> (se conforma el equipo de trabajo, se elabora la programación de actividades, se asignan responsables a cada una y se determinan los tiempos estimados para llevarlas a cabo)
Identificar la funcionalidad que se pretende alcanzar con el software	<u>Modelo de actores</u> (identifica los tipos de usuario que utilizarán el software y describe sus características) <u>Modelo de casos de uso</u> (establece un modelo general de las funciones que cubrirá el sistema a través de diagramas de casos de uso y su especificación)
Establecer los criterios de medición de calidad del proceso, considerando aspectos tanto técnicos como pedagógicos	<u>Modelo de aceptación</u> (incluye las características mínimas que deben cumplirse para que el producto se acepte)

### *Análisis y Diseño Inicial*

En la fase de análisis y diseño inicial se analiza el dominio del problema y se establece la arquitectura del sistema. En este punto se describen a detalle los requisitos del software y las características educativas y de comunicación que el producto debe contemplar. En la tabla 2 se detallan estas actividades.

### *Plan de iteraciones*

Una vez identificados los requisitos a cubrir con el software se procede a analizar cuántos subproductos funcionales pueden producirse de modo que se puedan liberar partes operativas del sistema final, con el objetivo de llevar un mejor control en el desarrollo. Una vez identificados los incrementos se priorizan y se colocan con mayor prioridad aquellos que cubren los conocimientos base. En la tabla 3 se muestran los resultados de esta fase.

**Tabla 2. Actividades y Artefactos de la Fase de Análisis y Diseño Inicial.**

<b>ACTIVIDAD</b>	<b>ARTEFACTO</b>
Identificar los requisitos funcionales y no funcionales que se cubrirán con el software	<u>Modelo de requisitos</u> (Se determinan los requisitos que debe cumplir el software en cuanto a funcionalidad, comunicación, interfaz y docencia.)
Establecer la arquitectura del software	<u>Descripción de la arquitectura</u> (establecer la arquitectura base sobre la cual se desarrollará el software; se debe considerar que dicha arquitectura sea capaz de atender adecuadamente las tareas de aprendizaje que se van a manejar)
Elaborar el diseño educativo	<u>Modelo educativo</u> (Se definen el objetivo terminal y los subobjetivos, y en base a éstos se establecen las tareas de aprendizaje apegadas al tipo de modelo educativo)
Elaborar el diseño de comunicación general del producto	<u>Modelo de interfaz</u> (diseño de las zonas de comunicación y pantallas que se seguirán a lo largo del desarrollo) <u>Modelo de navegación</u> (diseño de los caminos de navegación generales que se presentarán al usuario) <u>Prototipo de la interfaz de usuario</u> (establecer las plantillas de diseño que se seguirán a lo largo del desarrollo)

**Tabla 3. Actividades y Artefactos de la Fase del Plan de Iteraciones.**

<b>ACTIVIDAD</b>	<b>ARTEFACTO</b>
Diseñar las iteraciones de forma que las versiones ejecutables cubran objetivos didácticos bien planeados, de acuerdo a la secuencia de temas.	<u>Plan de iteraciones</u> (dividir el desarrollo en iteraciones, cuidando de que cada iteración cubre requisitos y objetivos educativos completos)
Priorizar las iteraciones, de modo que las que contienen conocimientos básicos que se requieren como base para aprendizajes posteriores se ejecuten primero.	<u>Lista de Iteraciones Priorizadas</u> (ordenar las iteraciones programadas de forma lógica de acuerdo a los contenidos)

### **Diseño Computacional**

Para cada iteración se debe elaborar el diseño computacional detallado, de modo que sirva de base para el desarrollo. Los artefactos y actividades propios de este paso se muestran en la tabla 4.

### **Desarrollo**

Se desarrolla en esta fase el producto, implementando la arquitectura de manera que se obtiene una versión del software lista para que sea utilizada por los usuarios finales. En la tabla 5 se incluyen sus elementos a detalle.

### **Fase de Despliegue**

En la fase de despliegue se realiza la transición del producto a los usuarios. Aquí se culmina con una versión ejecutable del producto. Las actividades y artefactos de esta fase se describen en la tabla 6. Al finalizar esta etapa se evalúa la conveniencia de continuar los desarrollos, y en su caso regresar a la etapa de diseño computacional para continuar con el siguiente incremento.

***Tabla 4. Actividades y Artefactos de la Fase de Diseño Computacional.***

<b>ACTIVIDAD</b>	<b>ARTEFACTO</b>
Realizar el plan de trabajo de la iteración	<u>Plan de trabajo</u> (se determinan las tareas que se realizarán en el diseño del software, se asignan a los miembros del equipo y se calendarizan)
Elaborar el diseño computacional	<u>Modelo de diseño</u> (detallar el diseño a través de diagramas de clases y secuencia, incluir la descripción de clases y métodos; para los desarrollos que requieren bases de datos, incluir la especificación de diccionario de datos y diagramas entidad-relación.)
Refinar el diseño de navegación	<u>Modelo de navegación refinado</u> (diseñar los caminos de navegación específicos para la iteración en desarrollo)
Refinar prototipo de interfaz	<u>Modelo de interfaz usuario</u> (desarrollar las pantallas específicas para los elementos de la iteración en desarrollo)

***Tabla 5. Actividades y Artefactos de la Fase de Desarrollo.***

<b>ACTIVIDAD</b>	<b>ARTEFACTO</b>
Desarrollar los componentes	<u>Modelo de desarrollo</u> (Determinar los componentes a desarrollar y documentarlos.)
Probar los componentes	<u>Modelo de pruebas unitarias</u> (Realizar pruebas de los componentes contra los criterios previamente establecidos. Estas pruebas deben incluir las pruebas del diseño instruccional)
Integrar al desarrollo previo	<u>Modelo de Integración</u> (establecer un plan para incorporar el nuevo desarrollo a la liberación previa si es el caso)
Realizar pruebas de integración	<u>Pruebas de integración</u> (realizar pruebas para verificar que la incorporación del nuevo incremento no ha inducido fallas al sistema)

*Tabla 6. Actividades y Artefactos de la Fase de Despliegue.*

<b>ACTIVIDAD</b>	<b>ARTEFACTO</b>
Entregar producto al usuario	<u>Producto</u> (Se debe entregar el producto debidamente empacado, etiquetado y con información sobre su contenido, aplicación, población objetivo y requerimientos de instalación) <u>Manual de Usuario</u> (Debe contener información detallada de cómo utilizar el software) <u>Manual de Instalación</u> (información de los requerimientos para su funcionamiento y procedimiento de instalación)
Evaluar las características de calidad y satisfacción de los usuarios	<u>Aceptación del Usuario</u> (realizar pruebas con los usuarios finales y comprobar su grado de satisfacción y efectividad del software)
Evaluar la conveniencia de continuar con otro incremento al producto	<u>Evaluación de despliegue</u> (analizar los resultados de la prueba de aceptación del usuario y determinar si es conveniente seguir con otra iteración.)

### Prototipo

Se elaboró un prototipo que cubre la primera iteración de una aplicación para el aprendizaje de factorización. El título del proyecto es UNIMAT y está dirigido a alumnos de nivel medio superior. A continuación se muestran algunos de los artefactos desarrollados.

En la figura 2 se ilustra el Modelo Instruccional generado en la Fase Conceptual, en el cual se describen los objetivos didácticos que debe cubrir el software, las fuentes bibliográficas que apoyan dichos objetivos, el modelo educativo a usar y la forma de evaluación.

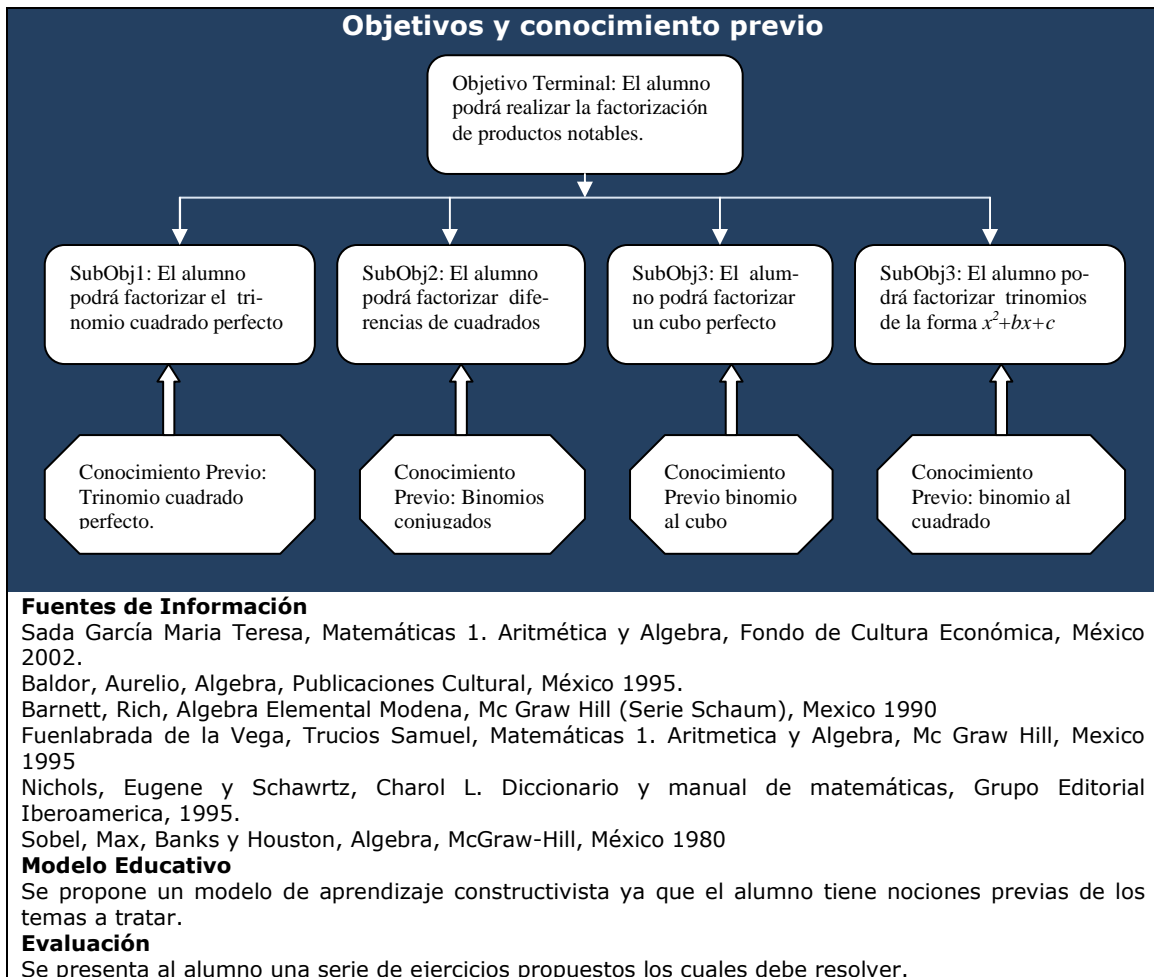
En la figura 3 se muestra el modelo de requisitos de la fase de análisis y diseño inicial. En este punto se describe la forma como se atenderá el requerimiento didáctico a través del software.

En la figura 4 se muestran dos pantallas del producto final: la pantalla de bienvenida al sistema y una pantalla que presenta el material de estudio de la factorización de trinomio cuadrado perfecto.

### Resultado

La metodología que se propone en este artículo, MeISE, se basa en un modelo de ciclo iterativo, toma de los modelos analizados la idea de incluir aspectos pedagógicos y comunicacionales, además de mecanismos de ingeniería de software. Además enriquece el modelo al segmentarlo en dos etapas: una en donde se establece la estructura general del sistema y se divide el desarrollo en una serie de incrementos, y otra donde se detalla el diseño de cada incremento para desarrollarlo e integrarlo al producto final. Propone además las actividades y productos a generar en cada fase.

La prueba de la metodología a través del desarrollo de un prototipo, arrojó una buena aceptación por parte del equipo de desarrollo. El producto final se presentó a profesores de la materia de álgebra a nivel medio superior de quienes se obtuvieron comentarios aceptables. Se realizará una prueba con alumnos en el próximo ciclo escolar para evaluar el beneficio didáctico de dicho software.



**Figura 2. Modelo Instruccional del software UNIMAT.**

Nombre del Requerimiento		Trinomio Cuadrado Perfecto					Número	1
Tipo	Descriptivo	X	Gráfico		Númerico	X	Sonoro	
Objetivo de aprendizaje asociado				Subobjetivo1				
Objetivos del SE:								
<ul style="list-style-type: none"> <li>o Proporcionar material relacionado con la factorización del trinomio cuadrado perfecto.</li> <li>o Dar la posibilidad de revisar la información sobre el concepto de trinomio cuadrado perfecto.</li> <li>o Presentar una serie de ejercicios resueltos.</li> <li>o Presentar ejercicios para evaluación.</li> </ul>								
Relaciones		Conocimientos previos de trinomio cuadrado perfecto						
Flujo Normal:								
<p>El sistema presenta una serie de páginas consecutivas que muestran la información sobre el tema. En primer lugar los conceptos básicos, luego ejemplos y finalmente la evaluación.</p> <p>En la evaluación se presentan en forma aleatoria 10 ejercicios que el alumno debe contestar, si responde correctamente 7 se registra como acreditado el tema guardando su calificación y dando acceso al siguiente tema.</p>								

**Figura 3. Parte del Modelo de Requisitos de UNIMAP.**

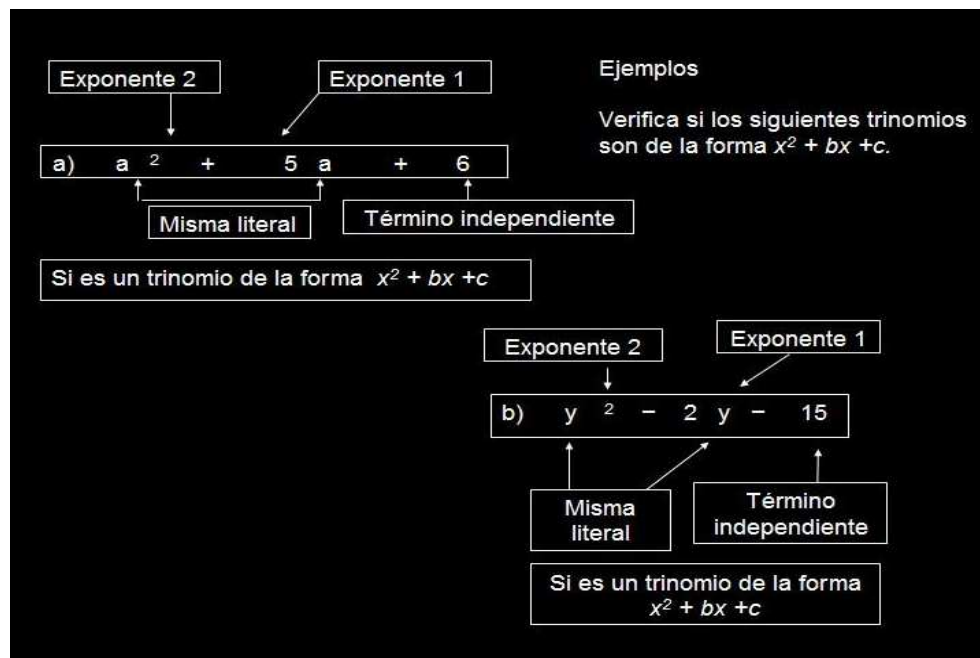


Figura 4. Pantalla de bienvenida (arriba) y pantalla de trinomio cuadrado perfecto.

## Conclusiones

Se presenta la metodología MeISE para el desarrollo de software educativo basada en un modelo iterativo, que contempla aspectos orientados a obtener un producto de calidad desde los puntos de vista técnico y didáctico, para lo cual incluye aspectos de diseño pedagógico y de interfaz humano-computadora que ayudan a asegurar la efectividad didáctica del software. La principal aportación de MeISE es que divide el proceso del ciclo de vida en dos etapas. Primeramente está la de definición, en la que se establecen la definición completa del contenido del producto, el modelo didáctico que se utilizará, las estructuras de comunicación adecuadas al mismo así como un plan de desarrollo que lo divide en iteraciones que deben terminar en un producto completo desde el punto de vista educativo con objetivos bien definidos. La segunda parte es la etapa de desarrollo, en la cual

se trabaja iteración por iteración realizando el diseño computacional, se desarrolla el producto y se entrega al usuario para su uso. Esta separación de actividades permite, antes de iniciar el desarrollo, conceptualizar en forma completa y clara los objetivos pedagógicos que cubrirá el producto así como elegir la estrategia didáctica más conveniente al caso a tratar, con lo que se puede asegurar la calidad educativa del software. De esta forma, una vez definido el producto a desarrollar, la segunda fase enfoca los esfuerzos al desarrollo utilizando un enfoque iterativo e incremental que permite un mejor control y asegura la calidad en el proceso de ingeniería de software.

## Referencias

Díaz-Antón, M.G., Pérez, M.A., Grimmán, A.C. y Mendoza, L.E. "Propuesta de una metodología de desarrollo de software educativo bajo un enfoque sistémico de calidad", Trabajo para Obtener Especialidad, Universidad Simón Bolívar, Caracas, Venezuela, 2002, <http://www.academia-interactiva.com/ise.pdf>, consultada el 14 de enero de 2008.

Cataldi, Z., Lage, F., Pessacq, R. y García Martínez, R., "Metodología extendida para la creación de software educativo desde una visión integradora", Revista Latinoamericana de Tecnología Educativa, Vol. 2, No. 1, 2006, [http://www.unex.es/didactica/RELATEC/Relatec\\_2\\_1/cataldi\\_lage\\_2\\_1.pdf](http://www.unex.es/didactica/RELATEC/Relatec_2_1/cataldi_lage_2_1.pdf), consultada el 7 de diciembre de 2007.

Galvis, A., "Ingeniería de software educativo", Ediciones UNIANDES. Colombia, 2000.

GIDSE, Grupo de investigación y desarrollo de software educativo, "Consideraciones sobre el proceso de autoría de aplicaciones hipermedia", 2006, <http://gidse.univalle.edu.co>, consultada el 14 de enero de 2008.

Gómez Castro, R., Galvis Panqueva, A., y Mariño Drews, O., "Ingeniería de software educativo con modelaje orientado por objetos: un medio para desarrollar micromundos interactivos". Revista de Informática Educativa. Vol 11, No. 1, pp. 9 - 30, 1998

Hinostroza, E., Hepp, P., y Straub, P., Un método de desarrollo de software educativo. Revista de Informática Educativa, Vol 9, No. 1, pp. 9 -32, 1996.

Peláez G. y López, B., "Metodología para el desarrollo de software educativo, DESED", Revista UPIICSA, Vol XIV y XV, 2006, México.

Salcedo Lagos, P, "Ingeniería de Software Educativo, Teorías y Metodologías que la Sustentan", Revista Ingeniería Informática, Edición 6, 2000, <http://www.inf.udec.cl/revista/edicion6/psalcedo.htm>. Consultada el 17 de enero de 2008.

Sierra, E., "Sistemas tutoriales inteligentes centrados en reparación de mecanismos. Una propuesta metodológica de diseño", Revista de Informática Educativa y Medios Audiovisuales, <http://www.fi.uba.ar/laboratorios/lie/Publicaciones/100.pdf>, Marzo 2007. Consultada el 17 de enero de 2008.