Propuesta metodológica para el desarrollo de software educativo en la Universidad de Holguín

Carlos J. Madariaga[1], Yasnalla. Rivero[2], Arquimedes R. Leyva[1]

[1]Direccion de informatización – Universidade de Holguín
Caixa Postal 15.064 – 91.501-970 – Holguín– Cuba

[2]Facultad de Informática-Matemática – Universidade de Holguín
Caixa Postal 15.064 – 91.501-970 – Holguín– Cuba

cmadariaga@facinf.uho.edu.cu, yasnalla@facinf.uho.edu.cu,
aleyvat@facinf.uho.edu.cu

Abstract. *For efficient use of information technologies and communication in the teaching-learning process at the University of Holguin, among other things requires a methodology for developing educational software. Based on this need has led research that has examined the theories of contemporary learning the characteristics of the processes of software development, traditional models of software development, the agile methodologies and proposes a methodology for developing educational software adapted to the characteristics and needs of teaching at the University of Holguin. This proposal describes a set of steps that guide the development process of software products that support the teaching-learning process.*

Resumen. *Para el uso eficiente de las tecnologías de la informática y las comunicaciones en el proceso enseñanza aprendizaje, en la Universidad de Holguín, se requiere entre otros elementos una metodología de desarrollo de software educativos. Basados en esa necesidad se ha conducido una investigación que ha analizado las teorías del aprendizaje contemporáneo, las características de los procesos de desarrollo de software, los modelos tradicionales de desarrollo de software, las metodologías agiles y propone una metodología de desarrollo de software educativo adaptada a las características y necesidades de la docencia en la Universidad de Holguín. Dicha propuesta describe un conjunto de fases que permiten guiar el proceso de desarrollo de productos de software que apoyen el proceso de enseñanza aprendizaje.*

1. Introducción

Respecto al diseño de software educativo, se puede aseverar que de la propia etimología del termino se desprenden los marcos conceptuales tenidos en cuenta en esta publicación los que esencialmente están centrados en dos pilares que son las teorías del aprendizaje y las metodologías propias de la ingeniería de software, arribando a lo que denominamos ingeniería del software educativo.

Como punto de partida, se realizó una prospección de las teorías del aprendizaje contemplando el condicionamiento operante de Skinner (1958-63), la instrumentalización cognitiva de Bruner (1988-1991), la perspectiva psicogenética de Piaget (1989), el aprendizaje significativo de Ausubel (1973), los mapas conceptuales de Novak (1984), el marco sociocultural de Vigotzkii (1989), la teoría uno de Perkins (1995), la teoría de las inteligencias múltiples de Gardner (1987, 1993,1995), sin dejar de lado los aportes de Coll (1994), Sancho (1994), Jonhson-Laird (1998), Pozo Municio (1998), Cabero (1992-2000), a fin de considerar de acuerdo a las necesidades tanto el conductismo, el constructivismo como las teorías cognitivas actuales y sus variantes, haciendo énfasis en conductismo positivista de la teoría de Vigotzki que es la generalizada en Cuba .

Estas teorías permitirán elaborar el software de acuerdo a la necesidad en cada caso, ya sea un programa para el entrenamiento de habilidades técnicas de los estudiantes, de ejercitación, de refuerzo o para la construcción de significados a partir del descubrimiento. Es decir de acuerdo al requerimiento el docente podrá utilizar la que crea más conveniente.

Igualmente fueron analizados los dos procesos que intervienen en el desarrollo del software educativo; la gestión del proyecto y la ingeniería de software, Royce (1970); Brooks (1975) ; Mantei (1981); Martin (1991); Constantine (1993); Ken & Hunter (1994) ; Whitaker (1994); Butler (1995); Hanna. (1995); Bohem (1996) Council (1999); Reel (1999); Pressman (2009). Dentro de la gestión de proyectos de software fueron identificados los procesos de gestión del personal, del producto, del proceso y del proyecto. Mantei (1981); Constantine (1993); Whitaker (1994); Bohem (1996) Council (1999); Reel (1999); Pressman (2009).

Casi todos los desarrolladores de software educativo reconocen que la elección del modelo de desarrollo va acorde con las características del entorno donde este va a ser desarrollado e implementado Toth (2005); Sommerville (2007); Costa (2010).

Una vez analizados los modelos tradicionales de desarrollo de software (Modelo Lineal Secuencial, Modelo de construcción de prototipos, Modelo de desarrollo rápido de aplicaciones, Modelo evolutivo incremental, Modelo evolutivo en espiral). Y enfocados en el desarrollo de software educativo siguiendo las etapas de dichos modelos podemos concluir que:

- A menudo es difícil que el cliente del software educativo exponga explícitamente todos los requisitos antes de comenzar el proyecto. Y tiene dificultades a la hora de acomodar la incertidumbre natural al comienzo de muchos proyectos.
- El cliente del software debe tener paciencia. Una versión de trabajo del(los) programa(s) no estará disponible hasta que el proyecto esté muy avanzado. Un grave error puede ser desastroso si no se detecta hasta que se revisa el programa.
- En general están concebidos para proyectos grandes con requerimiento de mucho personal y documentación.

Por un lado para muchos equipos de desarrollo el uso de metodologías tradicionales les resulta muy lejano a su forma de trabajo actual. Además las características de los

proyectos para los cuales las metodologías agiles han sido específicamente pensadas se ajustan a un amplio rango de proyectos de desarrollo; incluyendo con especial énfasis a los equipos de desarrollo pequeños, con plazos reducidos, requisitos volátiles y basados en nuevas tecnologías cambiantes. Dicho sea de paso esta son características casi calcadas a las que presentan los proyectos de software educativos actuales en la universidad de Holguín.

2. Características de las metodologías ágiles

De manera general, las metodologías ágiles pueden explicarse según Meneses, A., Peñalver, G., Rodríguez, M., Fernández, R., & Pino, S. (2010), a través de los siguientes cuatro principios fundamentales.

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. Dado que el proceso de desarrollo es creativo, no es posible pensar que las personas funcionen respondiendo a órdenes o procesos rígidos.
- Desarrollar software que funciona más que conseguir una buena documentación. Puesto que si el software no funciona la documentación no vale de nada. A nivel interno puede haber documentación, pero solo la necesaria y a nivel externo lo que el cliente requiera.
- La colaboración con el cliente más que la negociación de un contrato. Supone que la satisfacción del cliente con el producto será mayor, mientras exista una conversación y realimentación continua entre este y la empresa.
- Responder a los cambios más que seguir estrictamente un plan. Puesto que si un proyecto de software no es capaz de adaptarse a los cambios fracasara, especialmente en productos de gran envergadura. La estrategia de planificación se basa en: planes detallados para las próximas semanas, planes aproximados para los próximos meses y muy generales para plazos mayores.

3. Breve síntesis de metodologías agiles más utilizadas en la actualidad.

Para una mejor comprensión de las características que poseen las metodologías agiles nos proponemos a continuación exponer las principales características de las más utilizadas por la comunidad de desarrolladores:

- Extreme Programming (XP)

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios.

- Scrum

Scrum define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se

muestra al cliente. La segunda característica importante son las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. Además SCRUM permite la autonomía de los equipos de trabajo. Utiliza reglas para crear un entorno ágil de administración de proyectos y no prescribe prácticas específicas de ingeniería.

- Crystal

Crystal Methodologies se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (Amarillo) (3 a 8 miembros) y Crystal Orange (Naranja) (25 a 50 miembros). El equipo ajusta continuamente sus convenciones de trabajo para adaptarse a:

- Las personalidades particulares del equipo
- El ambiente local de trabajo actual
- Las particularidades de las asignaciones específicas

La cantidad de detalle necesario en la documentación de los requerimientos, diseño y planeamiento varía según las circunstancias del proyecto. Quizás no sea posible eliminar todos los work products (productos del trabajo) y las notas promisorias inmediatas tales como los documentos de requerimientos, diseño y los planes para el proyecto, pero pueden reducirse a una comunicación corta, enriquecedora, e informal entre el equipo.

- Dynamic Systems Development Method (DSDM)

DSDM. Define el marco para desarrollar un proceso de producción de software. Nace en 1994 con el objetivo de crear una metodología RAD unificada. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases.

- Feature Driven Development (FDD)

FDD. Define un proceso iterativo que consta de 5 pasos. Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software.

Los principios de FDD son pocos y simples:

- Se requiere un sistema para construir sistemas si se pretende escalar a proyectos grandes.
- Un proceso simple y bien definido trabaja mejor.
- Los pasos de un proceso deben ser lógicos y su mérito inmediatamente obvio para cada miembro del equipo.

- Vanagloriarse del proceso puede impedir el trabajo real.
- Los buenos procesos van hasta el fondo del asunto, de modo que los miembros del equipo se puedan concentrar en los resultados.
- Los ciclos cortos, iterativos, orientados por rasgos (features) son mejores.

4. Propuesta

En nuestra propuesta, se usará SCRUM para la planificación de los proyectos que usarán métodos ágiles como metodología para su proceso de desarrollo pues SCRUM es una forma de gestionar proyectos de software, no es una metodología de análisis, ni de diseño, es una metodología de gestión del trabajo. Aquí no se pretende implantar dicha metodología en su totalidad, de la misma serán tomadas algunas prácticas para su implantación.

Para llevar a cabo el proceso de desarrollo del proyecto se tomará en cuenta las mejores prácticas de la metodología XP, procurando que el proceso sea efectivo y eficiente. La idea es producir rápidamente versiones del sistema que sean operativas, aunque obviamente no cuenten con toda la funcionalidad pretendida para el sistema pero sí que constituyan un resultado de valor para el negocio.

1. Cada entrega es lo más corta posible:

- Contenga requisitos más valiosos del sistema (básicos).
- Reducen el riesgo: mayor retroalimentación desde el cliente, y más frecuente.

2. Minimizar el número de historias de usuarios que componen una entrega:

   No realizar historias de usuarios a medias.

3. A cada programador se le asigna una tarea de la historia de usuario.

Esta sección tiene como objetivo articular diferentes tópicos de ingeniería de software como son el enfoque clásico, los enfoques modernos y las metodologías de desarrollo ágil para proponer algunas orientaciones para el proceso de desarrollo de software educativo. Para cumplir con el objetivo se plantean una serie de fases o etapas por las cuales debe transitar el proceso de desarrollo de software educativo, entre las cuales proponemos:

- Planeación
- Análisis
- Diseño & Implementación
- Pruebas
- Evaluación

Para la mejor comprensión del esquema del modelo de la propuesta de este artículo así como la iteración de las fases del mismo se puede visualizar en la Figura 1.

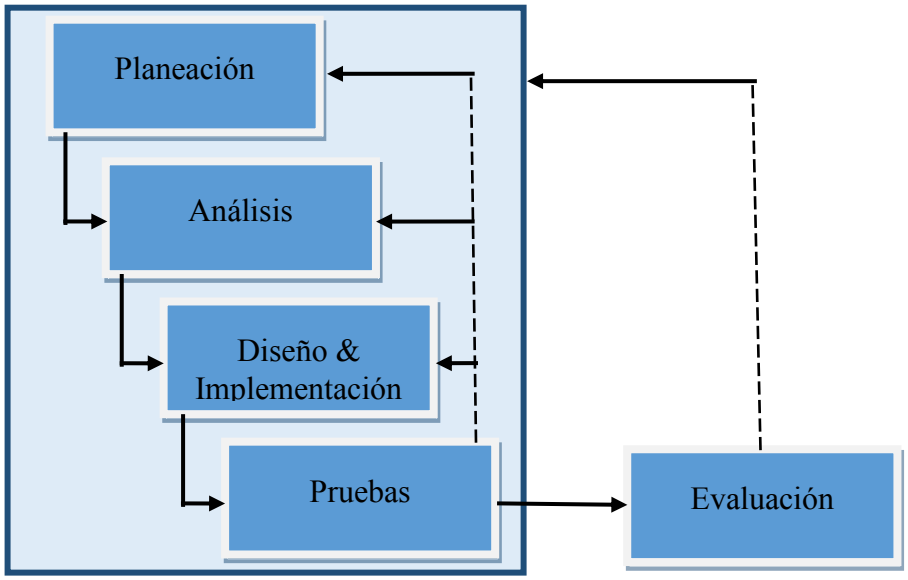

Figura 1. Esquema de la metodología propuesta.

Planeación

Durante la planeación se propone la conformación del equipo y las herramientas de trabajo, así como el establecimiento de las políticas de trabajo del proyecto; estas deben estar enmarcadas en su gran porcentaje al fortalecimiento de los procesos de enseñanza y aprendizaje y a la incorporación de las nuevas tecnologías en los ambientes educativos. Es vital establecer pautas de trabajo en equipo como son: la comunicación del equipo, la incorporación del docente y/o estudiante en el equipo, el énfasis en el desarrollo incremental a través de iteraciones cortas. Igualmente se definen aquí las pautas colaborativas con el cliente, en este caso la institución educativa, docente y/o estudiante. Para la conformación del equipo de trabajo se proponen los siguientes roles:

- Gerente de proyecto: Es el encargado de establecer, aplicar y controlar las políticas del proyecto. Así como modificar sobre la marcha del proceso los aspectos que así lo requieran.
- Analista: Es la persona encargada de construir el modelo de análisis del sistema de software. Además identifica los requerimientos que necesita el cliente en el sistema. Construye además la arquitectura del sistema junto con sus maquetas y rutas de navegación. Igualmente modificara sobre la marcha del proceso los aspectos que lo requieran.
- Diseñador: Es el especialista en el manejo de las medias que dará la estética del sistema acorde con lo definido en el análisis. Igualmente modificara sobre la marcha del proceso los aspectos que lo requieran.
- Programador: Es la persona encargada de implementar las funcionalidades del sistema. Modificará sobre la marcha del proceso los aspectos que lo requieran.
- Cliente: Es la persona que solicita el software, lo va a utilizar y probar

- Evaluador: Es la persona que determinara si es generalizable o no el software atendiendo sus características. Es una persona ajena al grupo de desarrollo del software.

En esta fase se definen los aspectos fundamentales sobre los que se desarrollara el software. Entre los que se proponen:

- Aspectos educativos. En este apartado se define el modelo pedagógico a seguir así como las estrategias didácticas que se le dará soporte con el software.
- Aspectos tecnológicos. Se describen características de los componentes de hardware y software necesarios para dar soporte al software educativo.
- Aspectos metodológicos. Se describe todas las actividades y estrategias que deben acompañar al uso del software educativo en el proceso de enseñanza aprendizaje.
- Aspectos organizacionales. Se detallan las actividades relacionadas con la gestión del proceso de desarrollo e implementación del software educativo.

Análisis

En esta fase se dividirá el trabajo en dos etapas. La primera dedicada al levantamiento de requerimientos y la segunda al modelo del sistema.

Un requerimiento es una característica que debe tener el sistema o una restricción que debe satisfacer para que sea aceptado por el cliente. La obtención de requerimientos se enfoca en la descripción de propósitos del sistema. El cliente los desarrolladores y los usuarios identifican los problemas y estos son modelados por el analista para construir funciones que el sistema posea para atacar a estos problemas identificados previamente.

Como producto de esta fase se debe producir un modelo de casos de uso acompañado de un documento donde se especifiquen los requerimientos no funcionales.

En metodologías tradicionales se trata de cumplir con requerimientos no funcionales sin que llegue a convertirse en el eje central del producto. Para la ingeniería del software educativo es de relevante importancia la determinación de requerimientos de escenarios y personajes debido a que estos elementos son el medio a través de los cuales el estudiante va a adquirir los conocimientos del aprendizaje

El análisis da como resultado un modelo del sistema que pretende ser correcto, completo, consistente y verificable. El cliente y el usuario están involucrados, por lo general, en esta actividad, en especial cuando se necesita cambiar la especificación del sistema y cuando se necesita recopilar información adicional. El análisis se enfoca en la producción del modelo del sistema (modelo de análisis). Como producto de esta fase se debe producir un modelo de clases si es necesario acompañado del refinamiento del modelo de casos de uso generado en la etapa anterior, estos modelos deben ser desarrollados solo a nivel de diagrama UML solo se deben especificar con mayor detalles en caso de que los clientes lo requiera.

Diseño & Implementación.

Esta fase se decide unificarla, y no dividirla en dos como casi todas las metodologías de desarrollo plantean, ya que no se considera que para el desarrollo de software educativo,

atendiendo a las características y complejidades del mismo, no se pueda implementar sobre la marcha del proceso de diseño las funcionalidades requeridas en dicho software.

El diseño de sistemas es la transformación del modelo de análisis en un modelo de diseño del sistema. Durante el diseño del sistema los desarrolladores definen los objetivos del diseño del proyecto y descomponen el sistema en subsistemas más pequeños que puedan ser realizados por equipos individuales.

En forma específica en esta etapa se debe generar un modelo de arquitectura e forma general y una lista de requerimientos no funcionales u objetos de diseño. Dentro del Diseño del sistema atendiendo a las regularidades del software educativo se requiere:

- Diseño educativo. Debe resolver los interrogantes que se refieren al alcance, contenido y tratamiento que debe ser capaz de apoyar el sistema. En este tipo de diseño se debe prestar especial atención al modelo pedagógico para diseñar las actividades del sistema de acuerdo al modelo o fusión de modelos sobre los cuales se quiere implementar el sistema y atendiendo al contexto en que se desenvuelve el usuario final del software.
- Diseño comunicacional. En este apartado se pretende especificar la forma como se comunica el usuario con el programa, estableciendo dispositivos y código o mensajes. Básicamente esta fase corresponde a lo que comúnmente se denomina diseño de interfaces
- Diseño computacional. En el diseño computacional se deben describir los usuarios del sistema complementados con funcionalidades a las cuales tienen acceso, para luego implementar módulos de acuerdo al diseño más apropiado que soporte los requerimientos funcionales y no funcionales del sistema. En este apartado se deben obtener los modelos funcionales, de clase y dinámico.

La implementación es el proceso mediante el cual se traducen lo modelos producidos en fases anteriores en código fuente. En forma paralela se deben integrar cada uno de los subsistemas desarrollados en forma segmentada.

Prueba

Las pruebas son el proceso de análisis de un sistema, o componentes del sistema, para detectar diferencias entre el comportamiento especificado y el observado. El objetivo de la prueba es aumentar la confiabilidad del sistema.

Esta fase no es inherente a un solo proceso del sistema, muy a menudo ocurre que las fallas en una parte del sistema (y su consecuente resolución) afecta el funcionamiento de otra e igualmente para su resolución se requiera un rediseño de funcionalidades y hasta de necesidades del cliente en vista de la experiencia durante el desarrollo. Esto es los que nos motiva a retroalimentar esta fase con cada una de las otras con el fin de introducir las mejoras a cada uno de los niveles que así lo requiere.

Evaluación

Es de vital importancia para los proceso de enseñanza aprendizaje determinar en la fase de requerimientos instrumentos de evaluación que puedan ser actividades dentro del sistema con los cuales se puedan medir el nivel de cumplimiento de los objetivos para cada uno de los contenidos que se pretenden desarrollar en el sistema. Se recomienda

atender a publicaciones precedentes de los autores de este artículo que atienden con mayor profundidad la temática en cuestión. Y tener en cuenta siempre en este punto al ser de vital importancia para la introducción del software educativo en la docencia:

- El instrumento de evaluación.
- Los evaluadores externos.
- Los tiempos de la evaluación.

5. Conclusiones

Los métodos agiles son una reacción a las formas tradicionales de desarrollo de software, admitiendo la necesidad de una alternativa al desarrollo de software educativo centrado en el proceso.

La propuesta metodológica para el desarrollo de software educativo describe un conjunto de fases que permiten guiar el proceso de desarrollo de productos de software que apoyen el proceso de enseñanza aprendizaje. Esta propuesta particularmente hace hincapié en los aspectos educativos, tecnológicos, metodológicos y organizacionales que permiten desarrollar un software educativo con una metodología ágil. Esta propuesta igualmente tiene entre sus características la adaptación a distintos tipos de proyectos de desarrollo de software educativo, es decir, a proyectos de distintos tamaños y características atendiendo las necesidades de la docencia universitaria.

References

Ausubel, D.; Novak, J. y Hanesian, H. (1978). Psicología educativa. Un punto de vista cognitivo. Trillas. Ediciones 1978, 1997.

Boehm, B. (1988). “Spiral Model for Software Developement and Enhancement". Computer. vol. 21, # 5, 1988, pp. 61-72.

Boehm, B. (1996). “Anchoring the Software Pocess”.IEEE Software. vol. 13, # 4, 1996, pp.73-82.

Boehm, B. (1998). “Using the WINWIN Spiral Model: A Case Study”. Computer. vol. 31, # 7, Julio 1998, pp. 33-44.

Brooks, F. (1975). The Mytical Man-Month. Addison Wesley.

Bruner, J. (1991). Actos de significado. Más allá de la revolución cognitiva.Madrid: Alianza.

Butler, J. (1995). Rapid Aplication Developement in Action. Managing System Development. Applied Computer Research, vol. 14, # 5.

Cabero, J. (1992). Diseño de Software Informático. Bordón, 44 (4), 383- 391.

Cabero, J. (2000). Tecnología Educativa. Madrid: Editorial Síntesis.

Coll, C. (1994). Psicología y Curriculum. Barcelona: Paidós.

Costa, A. P., Loureiro, M. J., & Reis, L. P. (2010). Metodologia Híbrida de Desenvolvimento Centrado no Utilizador. 5a Conferencia Ibérica de Sistemas e

Tecnologías de Información (CISTI2010) (pp. 192-197). Santiago de Compostela, España.

Hanna, M. (1995). Farewell to Waterfalls. Software Magazine.

Johnson-Laird, P. N. (1998). El ordenador y la mente: introducción a la ciencia cognitiva. Barcelona: Paidós.

Ken, J & Hunter, R. (1994). Inside RAD. McGraw-Hill.

Gardner, H. (1987-8). La nueva ciencia de la mente: Historia de la psicología cognitiva. Barcelona: Paidós.

Gardner, H. (1993). Las Inteligencias Múltiples: La teoría en la práctica. Barcelona: Paidós.

Gardner, H. (1995). La mente no escolarizada. Barcelona: Paidós.

Gilb, T. (1988). Principles of Software Engineering Management. Addison-Wesley.

Martin, J. (1991). Rapid Aplication Developement, Prentice-Hall.

McDermid, J. & Rook, P. (1993). "Software Development Process Models" en Software Ingineer's Reference Book.

Meneses, A., Peñalver, G., Rodríguez, M., Fernández, R., & Pino, S. (2010).Metodología ágil para proyectos de software libre. Revista cubana de ciencias informáticas. Universidad de ciencias Informáticas. Habana. Cuba

Parra, E. (2011). Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje – MESOVA. Revista Virtual Universidad Católica del Norte. ISSN-0124-5821

Pressman, R. (2009). Ingeniería de Software. Un enfoque práctico.

Royce, W.W. (1970). Managing the Developement of Large Software Systems: Concepts and Techniques.

Sancho, J. (1994). Para una Tecnología Educativa. Barcelona: Editorial Horsori.

Salazar, C., Reyes, R., Novas, W. (2011). Metodología para la evaluación de la calidad de los software educativos y su implementación. Recuperado el 10 de 08 de 2012, de XIV Congreso internacional de Informática en la Educación.: www.informaticahabana.cu/en/node/629

Svanaes, D., & Gulliksen, J. (2008). Understanding the Context of Design - Towards Tactical User Centered Design. Paper presented at the Nordic Conference on Human-Computer Interaction (NordiCHI2008), Lund, Sweden.

Skinner, B. F. (1958, 1963). Teaching Machines, Science, publicado en 1958; Reflection on a decade of teaching Machines, publicado en 1963, citados por Cruz Feliú, Jaime (1986) en Teorías del Aprendizaje y Tecnología de la Enseñanza, Trillas.

Sommerville, B. (2007). Software Engineering (Eighth Edition ed.): Addison Wesley.

Tchounikine, P. (2011) Computer Science and Educational Software Design. A Resource for Multidisciplinary Work in Technology Enhanced Learning. ISBN 978-3-642-20002-1

Toth, K. (2005). Which is the Right Software Process for Your Problem?

Vigotzki, L. (1978). Mind in Society. The development of higher psychological process. Cambridge, MA.: Harvard University Press.