¿Existe una situación de crisis del software educativo?

E. García Roselló, J. González Dacosta, M. Pérez Cota , J. B. García Pérez-Schofield,
V. G. Valdés Pardo

Resumen— **A pesar de la importancia cada vez mayor de los computadores y el courseware en la educación, siguen existiendo serios problemas en el desarrollo de software educativo. A partir del análisis de esta situación, en el presente trabajo se plantea la existencia de una crisis del paradigma de desarrollo del software educativo. La solución más prometedora, si no la única, pasaría por un desarrollo completo de la ingeniería del courseware, especialmente de la parte de esta disciplina más cercana a la ingeniería del software educativo, es decir, a las metodologías aplicadas en el proceso de desarrollo de los sistemas software que forman parte del courseware.**

***Palabras Clave*— Gestión del desarrollo de software, Ingeniería del software, Tecnología educacional.**

I. Problemas actuales del desarrollo de software educativo

El uso del computador como elemento integrante de procesos de enseñanza y aprendizaje se remonta a varias décadas atrás, y ha ido cobrando una creciente importancia, acentuada si cabe por la globalización de las comunicaciones y el acceso a la información proporcionado por la extensión de Internet y especialmente de la World Wide Web.

Así mismo, el desarrollo de *courseware* (entendido éste como una actividad formativa basada en o apoyada por computador) y de los sistemas basados en software educativo que lleva aparejados, no son una actividad nueva, sino que aparecen prácticamente al mismo tiempo que el uso de los computadores en la enseñanza. Podemos citar algunos ejemplos tempranos de lenguajes como Logo [1] , herramientas orientadas al *authoring* como Hypercard [2] o la anticipación del hipertexto [3] y de la hipermedia [4] [5] . Desde entonces hasta la actualidad se han producido multitud de aplicaciones, librerías, frameworks, componentes y entornos de desarrollo de software educativo.

Pero a pesar de todo ello, y aunque ha habido multitud de propuestas metodológicas sobre el desarrollo de todo tipo de courseware, que de hecho han llevado al reconocimiento de la ingeniería del courseware como disciplina de pleno derecho [6] , no ha habido un interés equiparable en el proceso de desarrollo del software con finalidad educativa que puede ser necesario construir como parte del courseware. Así, actualmente no se puede considerar que exista un conjunto de fundamentos, principios, métodos y herramientas claramente recogido, definido y ampliamente aceptado, que constituya la base de una completa ingeniería del software educativo.

Esto implica un claro vacío teórico y metodológico en la ingeniería del courseware. Por su propia naturaleza y por el tipo de productos que desarrolla, se trata de una ingeniería multidisciplinar [6] , donde un elemento significativo, a menudo el más importante, es el desarrollo de sistemas software.

Podemos concebir el núcleo primordial de la ingeniería del courseware como una fusión de otras dos disciplinas: el diseño instruccional por una parte, y la ingeniería del software por la otra, ya que, como es obvio pero a menudo no se valora suficientemente, el software educativo ***es* software** [6] [7] [8] .

Pero mientras que la aplicación del diseño instruccional al proceso de desarrollo de courseware ha recibido una enorme atención, la parte más centrada en el desarrollo de los sistemas software que sustentan el courseware, y que correspondería a la ingeniería del software, ha permanecido en un segundo plano. Se ha huido sistemáticamente de esta parcela de la ingeniería del courseware, del estudio de la aplicación concreta de la ingeniería del software al proceso de desarrollo de software educativo. Esto ha creado a veces la ilusión de que el desarrollo de software como parte del proceso de desarrollo de courseware es una tarea relativamente menor, que puede ser reducida a unos pasos predefinidos, encapsulada en herramientas de autor, o que puede ser tratada independientemente del resto del proceso de desarrollo de courseware.

Una consecuencia se puede ver en la existencia de lo que varios autores han venido recientemente a nombrar como el "patrón de fracaso" del software educativo [9] [10] [11] [12] [13] . Estos autores opinan que la manera de abordar el proceso de desarrollo de software educativo que hasta ahora ha prevalecido es el mayor impedimento para la explotación de todo el potencial real del uso de entornos basados en

Este trabajo ha sido apoyado en parte por el Instituto de Electrónica Aplicada de la Universidad de Vigo, a través de un proyecto conjunto de investigación, financiado por la Agencia Española de Cooperación Internacional.

E. García Roselló J. González Dacosta y M. Pérez Cota están en el Departamento de Informática de la Universidad de Vigo. Edificio Fundición, Lagoas-Marcosende s/n, 36200 Vigo, España. Teléfono: +34-986-814078; fax: +34-986-812180; e-mail: {erosello | jdacosta | mpcota}@uvigo.es.

J. Baltasar García Pérez-Schofield está en la E.T.S de Ingeniería Informática, Campus As Lagoas, s/n. 32005 Ourense, España. (e-mail: jbgarcia@uvigo.es).

V.G. Valdés Pardo está en la Facultad de Ingenierá Eléctrica, Universidad Central de Las Villas, Santa Clara, Cuba (e-mail: gvaldes@fie.uclv.edu.cu).

computadores en la educación. La solución más factible que señalan sería la adaptación e integración de los principios, métodos y herramientas de la ingeniería del software en el desarrollo de courseware, tarea que hasta ahora está, en el mejor de los casos, incompleta.

Se podría argüir que el desarrollo de una ingeniería del software educativo es superfluo o redundante, existiendo ya una ingeniería del software como tal, suficientemente sólida como para proporcionar la base para la construcción de una amplia variedad de sistemas software. Pero como se argumenta a lo largo de este trabajo, un enfoque de este tipo pecaría de ingenuidad. Es un error pensar que el software educativo es menos complejo que el software de gestión, que es el que indudablemente ha recibido más atención de parte de la ingeniería del software, o que el de otros dominios, que en cambio sí han desarrollado una ingeniería del software específica a su dominio (por ejemplo para el desarrollo de sistemas de tiempo real [14] , sistemas operativos [15] o aplicaciones basadas en Web [16]).

El software educativo abarca una gama de subdominios, tipos de sistemas, requerimientos e idiosincrasias lo suficientemente amplia y diversa como para justificar el estudio de los principios, métodos y herramientas concretos de la ingeniería del software que mejor se adapten a su desarrollo, y cómo particularizarlos a este ámbito. Como afirman De Diana y van Schaik "*el énfasis claro e inequívoco en el aprendizaje humano y la adquisición de conocimientos diferencian al software educativo* (de otros tipos de software)" [8] .

Esta falta de desarrollo de una ingeniería del software educativo, para su integración en la ingeniería del courseware, ha tenido y tiene serias consecuencias, no sólo en el proceso mismo de producción de software educativo, sino que ha repercutido en la concepción y consideración que como herramienta pedagógica se tiene de él. Podemos de hecho identificar tres problemas fundamentales que afectan actualmente al courseware como enfoque didáctico, en el marco de la tecnología educacional: (1) su falta de impacto educativo, (2) su elevado coste, y (3) la dificultad de estimación de los recursos necesarios para su producción. En las siguientes subsecciones tratamos con más detalle cada uno de ellos.

A. La falta de impacto del courseware en los planes de estudio

Los inicios de la computación instruccional están llenos de profecías sobre el gran potencial educativo de la enseñanza basada en computador [1] . Pero la mejora real apreciable en el aprendizaje y su impacto en la educación a distintos niveles ha sido hasta ahora bastante menos llamativa que lo augurado [17] . En los últimos años varios autores han alzado sus voces a este respecto, denunciando la falta de impacto que el courseware tiene en los sistemas educacionales .

Por ejemplo, Roschelle & Kaput reclaman la integración de las innovaciones en investigación educativa en un conjunto de herramientas prácticas basadas en computador que permitan soportar la reforma de los planes de estudio [12] [13] . Consideran que, frente a la existencia de lo que ellos denominan una arquitectura basada en "aplicaciones isla", se requiere la **integración escalable** como un elemento crítico de la tecnología educativa. El software educativo actual, tal y como es concebido y producido, falla en este objetivo, ya que a menudo suele consistir en un simple programa, desarrollado en unos meses a un año. El problema de este paradigma de desarrollo es que impide pasar de un enfoque localizado, a un enfoque que conlleve cambios curriculares mayores. El software educativo actual es localmente efectivo, pero globalmente fragmentario, demasiado caro, e incompatible para lograr una amplia aceptación. Además, no tiene la capacidad de adaptarse rápidamente y de forma eficiente en costes y tiempo, a requisitos cambiantes, cualidad ésta necesaria para impulsar cambios reales en los planes de estudio. La actual estructura de producción de software educativo estaría por tanto, abocada al fracaso en la consecución de este objetivo.

Otros autores también han argumentado que la integración entre módulos software y entre software y plan de estudios es crucial para la adopción a gran escala de la tecnología en la educación, y que de hecho es esta falta de integración la culpable en gran medida del fracaso parcial de la instrucción basada en computador, por cuanto no se han alcanzado los objetivos que inicialmente se preveían [9] .

B. La necesidad de reducir los costes de desarrollo

Incrementar el impacto del software educativo en los planes de estudio, como se mencionaba en el punto anterior, requiere entre otras cosas reducir drásticamente los costes de producción [13] . Se han realizado diversos estudios sobre el esfuerzo de desarrollo de courseware y se ha comprobado que el desarrollo de código supone la mayor parte de dicho esfuerzo, variando entre un 23% del tiempo total en tutoriales hasta un 30% en el desarrollo de simulaciones complejas [18] . Así mismo, el porcentaje medio del esfuerzo total de desarrollo de un courseware que requiere la implementación del software se situaría en un 48% [19] . Esto tiene serias implicaciones sobre el coste de desarrollo y mantenimiento del software.

Hoy en día a medida que el coste del hardware va siendo menor, el gasto real para desarrollar software educativo de buena calidad se ve incrementado por el riesgo de fracaso asociado a la producción de ese software [18] [20] [21] [22] . A pesar de todo, el uso de los computadores para adiestrar y educar está creciendo, en gran parte gracias a la popularidad de la Web. Esto también impulsa la demanda de software de mejor calidad pero a la vez más asequible, por tanto con costes de producción más bajos, que de momento no existe más que en cantidades insuficientes [17] .

C. La falta de un método de estimación fiable del coste

El problema anterior está asociado a la dificultad de estimar el coste de desarrollo del software educativo. La creciente demanda de courseware de calidad, tecnológicamente más complejo y por tanto más caro, ha generado la necesidad, por parte de los desarrolladores, tanto comerciales como de otros ámbitos, de realizar estimaciones realistas del esfuerzo

requerido para un proyecto de desarrollo de courseware [23] . Los trabajos y propuestas realizados sobre este aspecto han arrojado hasta ahora un éxito más bien moderado, y los estudios han mostrado que adolecen claramente de la suficiente rigurosidad y fiabilidad como para ser suficientemente útiles en la práctica [24] .

II. NECESIDAD DE UNA INGENIERÍA DEL SOFTWARE EDUCATIVO

A la luz de lo anteriormente expuesto, parece evidente que existe una falta de cumplimiento de los objetivos originalmente fijados para el software educativo. Por tanto, como se expone seguidamente, creemos que hay argumentos suficientes para plantear la existencia de una crisis del actual modelo de producción del software educativo, y la necesidad de procurar una solución, a través de un cambio de paradigma, que creemos que debe venir dado por el desarrollo de una disciplina de ingeniería del software educativo.

A. Estado de crisis del software educativo

Tras observar los problemas descritos en el punto anterior, creemos que se puede establecer un claro paralelismo entre la situación actual del software educativo y la llamada "crisis del software" [25] [26] , término acuñado en la NATO Software Engineering Conference de 1968. Esta situación de crisis se reconoció entonces como consecuencia de varios problemas detectados en el desarrollo de software, principalmente:

- La dificultad para estimar los recursos humanos, económicos y temporales necesarios para los proyectos de desarrollo de software.
- La imposibilidad de dar respuesta en tiempo y forma adecuados a la creciente demanda de software cada vez más complejo.
- La dificultad y costes cada vez mayores del mantenimiento del software una vez desarrollado.
- La falta de calidad, fiabilidad y flexibilidad del software desarrollado.

Vemos que existe una clara coincidencia con la problemática actual del software educativo. En consecuencia, podemos afirmar que actualmente existe una **crisis del software educativo**, caracterizada principalmente por los siguiente problemas:

- Un excesivo coste de desarrollo, tanto en recursos como en tiempo, a menudo difícil de estimar previamente.
- Una calidad del proceso de desarrollo así como del producto final en muchos casos deficiente, y en cualquier caso difícil de estimar a priori y de controlar durante el proceso de desarrollo.
- Una falta de capacidad para adaptarse a requerimientos cambiantes, de forma suficientemente rápida y eficiente.

Resulta de todas formas llamativo que hasta ahora no haya habido una clara consideración acerca de un estado manifiesto de crisis en el software educativo. Si es bien cierto que, como hemos citado, numerosos autores han señalado la existencia de serios problemas de fondo, no ha habido una constatación ampliamente consensuada de la situación.

Los motivos que sustentan esta falta de constatación de la existencia de una crisis se asocian a la propia naturaleza del software educativo y de su contexto. Por ejemplo, buena parte del software educativo no supone, o al menos no se percibe, como una ventaja competitiva decisiva para las instituciones que lo desarrollan o disponen de él, como de hecho ocurre en el entorno empresarial [27] . Al no existir una fuerte competitividad, el estímulo para el desarrollo de software de calidad es menor.

Por otra parte, el *authoring* siempre ha sido un paradigma importante en el software educativo, orientado a soportar sobre todo la posibilidad de que los usuarios profesores sin profundos conocimientos de informática y programación pudieran desarrollar aplicaciones por sí mismos y a su medida. En muchos casos, la consecuencia es que las aplicaciones que se abordan y desarrollan son relativamente pequeñas, implican una evolución limitada, y por lo mismo tienden a ser simplificadas. El resultado es muy a menudo un software educativo que no siempre alcanza la calidad adecuada para ser distribuído, y tendrá muchas veces un impacto limitado [28] .

Se ha minusvalorado el proceso de desarrollo de software educativo como parte de la producción de courseware. La insistencia en la tesis de que el software educativo sea desarrollado por los propios educadores ha llevado a la predominancia del desarrollo basado en herramientas de generación de aplicaciones y entornos cuya complejidad ha sido reducida intencionalmente para facilitar su uso por profesionales no informáticos, lo que ha conllevado irremediablemente una disminución de su potencia y flexibilidad de generación de software [29] .

El desarrollo de software educativo se debe considerar algo más que preparar la composición de una serie de textos, vídeos, imágenes, sonidos, etc..., enlazarlos y empaquetarlos para dar lugar a un curso. Un diseño estratégico del courseware debe considerar también como implementar ese software educativo de forma sistemática [30] [31] [32] .

La consideración de la existencia de un patrón de fracaso o crisis del software educativo aparece de hecho cuando se plantea la necesidad de aumentar su presencia e influencia en la enseñanza [13] . Esto implica el desarrollo de aplicaciones complejas, que a menudo no pueden ser implementadas con sistemas de authoring. El desarrollo de software educativo que pueda alcanzar los objetivos e impacto que cada vez más se reclaman, requiere de proyectos de mediano y gran tamaño, si no queremos vernos limitados a un software educativo de utilidad puramente personal. Este tipo de proyectos ha de ser un esfuerzo de equipo, multidisciplinar, y requiere de la participación directa de ingenieros informáticos. Por ello, y en clara analogía con la solución propuesta, en su momento, a la crisis del modelo genérico de producción de software, en la siguiente sección proponemos como única solución consistente a esta crisis del software educativo, la necesidad de desarrollar una ingeniería del software educativo, que integrada dentro de la ingeniería del courseware, proporcionaría el fundamento teórico y metodológico

requerido para el desarrollo de software de calidad de forma eficiente y sistemática dentro del proceso de producción de courseware.

B. Planteamiento de una ingeniería del software educativo

Establecido el paralelismo entre la históricamente llamada crisis del software y la situación actual del software educativo, parece lógico tratar de llevar esta similitud más allá, para comprobar si las soluciones apuntadas en el primer caso pueden ser aplicables al contexto educativo. En el estudio sobre la situación de crisis general del software, desde el primer momento se perfiló que la solución pasaba por el desarrollo de una *ingeniería* del software que aportara al proceso de desarrollo una aproximación más sistemática, en base a fundamentos y metodologías semejantes a los existentes en otras ingenierías tradicionalmente más desarrolladas [33] . Esta ingeniería debería llevar en última instancia a un modelo de desarrollo de software basado en componentes reusables, tendencia que se considera generalmente un signo de madurez de una ingeniería [34] [35] [36] .

También en el courseware, la solución que se ha apuntado a la crisis se centra en el paso de un paradigma de construcción *intuitiva* o *artesanal* del courseware a un modelo sistematizado [6] . De hecho, el diseño instruccional de courseware ha recibido una enorme atención [37] [38] . Pero en cambio, y como se ha comentado en A, no ha habido apenas avances sistemáticos en el área encaminados a la adquisición de un fundamento metodológico que condujera al desarrollo de una *ingeniería del software educativo*, que, recordémoslo, constituye, junto al diseño instruccional, la columna vertebral del desarrollo de courseware.

Es cierto que algunos autores ya han tratado de introducir la ingeniería de software en el desarrollo de courseware, proponiendo una diferenciación más clara entre la parte de diseño instruccional y la de desarrollo de software [28] .Tampoco han faltado propuestas significativas en el área iberoamericana, en lo referente a la producción de materiales educativos computarizados por parte de equipos en que participen tanto pedagogos como informáticos [39] . Por otra parte, muchas metodologías y paradigmas de ingeniería del software han atraído la atención de algunos autores por su potencial aplicabilidad en el software educativo. Por ejemplo, la orientación a objetos [40] [41] ha sido vista como un paradigma que facilitaría el diseño de software educativo y aumentaría su reutilización. Las tecnologías basadas en Internet han sido ampliamente explotadas en el desarrollo de courseware [42] . Así mismo, el uso de la tecnología basada en componentes ha atraído recientemente una gran atención. La existencia de una incipiente ingeniería del software basado en componentes [35] [36] , arquitecturas distribuídas como DCOM [43] , JavaBeans [44] o CORBA [45] así como de un bagaje de investigación importante en este campo a nivel del software en general que muestra sus notables ventajas, han llevado a varios investigadores en tecnología educativa a interesarse por este paradigma, presentándolo incluso como el más prometedor en la solución de muchos de los problemas del software educativo [10] [13] [46] .

Pero creemos que así como es innegable la existencia de potenciales ventajas de estas y otras tecnologías para el desarrollo de software educativo, sería precipitado plantearse su aplicación sin simultáneamente realizar una traslación adecuada, a la ingeniería del courseware, de los fundamentos y conceptos de ingeniería de software que implícitamente conllevan. Por ejemplo, creemos que sería necesario desarrollar una *ingeniería del software educativo basada en componentes*. La tecnología y el desarrollo de software basados en componentes no son simples técnicas que se pueden adoptar en cualquier contexto, sino que tienen importantes implicaciones a nivel estructural no sólo en el proceso de desarrollo de software, sino en las organizaciones mismas [36] . En el caso de la educación, esto supondría un cambio no sólo en la forma en que se desarrolla courseware, sino en el modo mismo en que éste se concibe, se utiliza e influye en el proceso educativo, y por tanto en la forma misma en que actúan y se relacionan los miembros de la comunidad educativa.

El acercamiento puntual y poco sistemático a ciertos principios y técnicas de ingeniería de software que se ha ido produciendo de forma continuada por parte de la comunidad de desarrollo de courseware no es suficiente ni adecuado. No puede solventar los problemas existentes, ni llevar al florecimiento de un verdadero cuerpo de conocimiento que complemente el ya existente, centrado en el diseño instruccional del courseware, con una ingeniería del software educativo, con objeto de sustentar de forma plena y efectiva la ingeniería del courseware.

Por tanto, creemos que es ineludible el planteamiento y desarrollo de una ingeniería del software educativo como disciplina de pleno derecho. A continuación, expondremos los requerimientos básicos para una ingeniería del software educativo.

C. Definición y requisitos para una ingeniería del software educativo

La ingeniería del software educativo, será, obviamente, una modalidad de ingeniería del software, y por tanto, podemos presentar una primera definición de ingeniería del software educativo simplemente adjetivando adecuadamente una definición clásica de ingeniería del software:

> "La ingeniería del software *educativo* es una disciplina que comprende todos los aspectos de la producción de software *educativo*, desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza." [47]

Si comparamos esta definición con la propuesta por Goodyear para la ingeniería del courseware [6] , podemos apreciar claramente la diferencia de ámbito entre ambas; mientras la primera se centra en el desarrollo de software educativo, la última abarca toda la producción de courseware, y de hecho ni si quiera menciona de forma explícita el software. Por tanto, se puede definir una integración entre ambas, sin que se

plantee un conflicto de competencias. De esta forma, podríamos refinar la definición anterior de la siguiente manera:

> "La ingeniería del software educativo es una disciplina que forma parte de la ingeniería del courseware, que comprende todos los aspectos de la producción de software educativo dentro del proceso de desarrollo de courseware, desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza."

Puesto que la ingeniería del software educativo se centraría en los aspectos de desarrollo de software como parte de la producción del courseware, es evidente que la otra componente fundamental de la ingeniería del courseware, como ya se ha comentado, sería el diseño instruccional. La ingeniería del courseware se podría definir en términos de la integración de estas dos disciplinas. Considerándola como una ingeniería del software aplicada al dominio específico del courseware, se pueden plantear los siguientes requerimientos básicos a la ingeniería del software educativo:

- Ofrecer principios, métodos y herramientas que soporten el desarrollo sistemático de software educativo de calidad, en todas sus etapas. Esto implica el desarrollo de modelos de proceso, metodologías y herramientas específicas para el software educativo, aprovechando sus características específicas frente a otros tipos de software.
- Disponer de procedimientos fiables de estimación de recursos económicos, tecnológicos, temporales y humanos para el desarrollo de software educativo.
- Conllevar una reducción de los costes y aumento de la calidad del producto software frente a estrategias de desarrollo no sistemáticas o *artesanales*.
- Como tendencia general de cualquier ingeniería, la optimización del proceso de desarrollo mediante la maximización de la reutilización.

Estos requerimientos se deducen de la simple particularización al ámbito del courseware de los que se plantean para una ingeniería del software en general.

Para plantear el desarrollo de las bases de una ingeniería del software educativo, es necesario abordar numerosas tareas con el objeto de poder dar respuesta a los requerimientos anteriormente expuestos. Algunas de ellas podrían ser:

- Caracterizar el software educativo, poniendo de relieve las peculiaridades de éste frente a otros tipos de software, para permitir particularizar la ingeniería del software a estas propiedades concretas. Si es posible, sería ventajoso disponer de métricas que permitan cuantificar algunas de estas características diferenciadoras.
- Así mismo, caracterizar el proceso de desarrollo de courseware frente a otros procesos de desarrollo existentes en otras áreas y para otros productos.
- Definir un marco de referencia para el ciclo de vida de courseware, es decir, qué tareas básicas y documentación se considerarán dentro de cualquier proyecto de desarrollo de courseware, y cómo se integrará en ella el proceso de desarrollo del software educativo. Esto establecerá el marco de aplicación de la ingeniería del software educativo y las fases para las que se han de proveer métodos, procedimientos y herramientas. Así mismo permitirá definir más claramente un conjunto integrado de documentación a generar, cubriendo tanto aspectos metodológicos instruccionales como de desarrollo de software.
- Tratar de modelar el dominio del courseware. Aunque esta tarea se perfila excesivamente ardua, sí tiene interés realizar modelos de dominios particulares del courseware, ya que pueden proporcionar mucha y valiosa información sobre cómo enfocar el proceso de desarrollo de software educativo en un dominio. Así mismo, el análisis del dominio es un prerrequisito para abordar la introducción de mecanismos de reutilización sistemática de componentes en un proceso de desarrollo [36] [48] . El análisis de dominios puede ser horizontal, orientado al modelo instruccional, o vertical, orientado al campo objeto de la instrucción.

Creemos que estas tareas exigen el esfuerzo de equipos interdisciplinares, formados principalmente por educadores, expertos en diseño instruccional, desarrollo de materiales de enseñanza y courseware, e ingenieros de software.

III. CONCLUSIONES

En este trabajo hemos expuesto una serie de hechos que nos han llevado a argumentar la existencia de una situación de crisis del software educativo, similar en muchos aspectos a la que en su momento se manifestó en la industria del software. El software educativo presenta una serie de retos y de problemas que sólo podrá resolver mediante un cambio de su actual paradigma de desarrollo, el cual parece abocado a lo que algunos autores han denominado un "patrón de fracaso". Este cambio, a su vez, exige el desarrollo de una -- todavía incompleta – disciplina de ingeniería del courseware, especialmente en lo que se refiere a la ingeniería del software educativo. Con este objeto, hemos tratado, en este trabajo, de ofrecer una definición y una visión general de los requerimientos que debería tener esta ingeniería, así como de las tareas a abordar para su desarrollo.

REFERENCIAS

[1] Papert, S. *Mindstorms*. New York, NY: Basic Books. Harper Colophon Books, 1980

[2] Apple Computer. *HyperCard* [computer software]. Cupertino: Apple Computer, Inc., 1987.

[3] Bush, V. "As We May Think". The Atlantic Monthly, July 1945, Vol. 176, #1, pp. 101-108.

[4] Engelbart, D.C. and English, W.K. "A Research Center for Augmenting Human Intellect". AFIPS Proceedings, Fall Joint Computer Conference, 33, 1968, pp. 395-410.

[5] Nelson, T. *Computer Lib/Dream Machines*. Redmond, WA: Microsoft Press, 1987.

[6] Goodyear, P. *Infrastructure for courseware engineering*. In: R.D. Tennyson and A.E. Barron, Editors Automating instructional design: computer-based development and delivery tools. Berlin: Springer-Verlag, 1995, pp. 11-31.

[7] Bostock , S. "Courseware Engineering - an overview of the courseware development process". Keele University, UK. 1998. Available on-line at http://www.keele.ac.uk/depts/cs/Stephen_Bostock/docs/atceng.htm

[8] De Diana, I. and van Schaik, P. "Courseware engineering outlined: an overview of some research issues". ETTI 30, 3 , 1993, pp.191-211.

[9] Bork A. " Why Has the Computer Failed in Schools and Universities?" J. of Science Education and Technology, 2(4), 1995, pp. 97-102.

[10] DiGiano C., Roschelle J. "Rapid-Assembly Componentware (RAC) for Education". In Proceedings of the International Workshop on Advanced Learning Technologies at Palmerston North, New Zealand. IEEE Computer Society Press, Los Alamitos, CA, 2000, pp. 37-40.

[11] Roschelle J., DiGiano C., Koutlis M., Repenning A., Jackiw N., Suthers D." Developing educational software components". IEEE Computer, 32:5-58, 1999.

[12] Roschelle J., Kaput J., Stroup W., Kahn T.M. "Scaleable Integration of Educational Software: Exploring the Promise of Component Architectures". J. of Interactive Media in Education, 98 (6), 1998.

[13] Roschelle J., Kaput J. "Educational Software Architecture and Sy stemic Impact: The Promise of Component Software". Journal of Educational Computing Research, 14(3), 217-228, 1996.

[14] Schiebe, M, Pferrer, S *Real-time systems – engineering and application*, Kluwer Academic Publishers, Dordrecht-London, 1992.

[15] Tanenbaum, A. S *Operating Systems: Design and Implementation*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1987.

[16] H.-W. Gellersen; M. Gaedke: "Object-Oriented Web Application Development". IEEE Internet Computing. Vol. 3, No. 1, Jan/Feb, 1999, pp. 60-68.

[17] Alessi S. M., Trollip S.R. *Multimedia for learning*. Allyn&Bacon, Massachusetts, third edition, 2000.

[18] Jay J., Bernstein K., Gunderson S. "Estimating computer-based training development times". ARI technical report 765, October 1987, Alexandria Research Institute for Behavioural and Social Sciences.

[19] Senbetta G. "An inquiry of time and cost estimating for computer-based training courseware design and development as determined by modified delphi method". Ph.D. thesis disertation, Purdue University, 1991.

[20] Hobbs, P. and Price, S. "Educational project management. Repeat after me". ALTC 94, Enabling Active Learning,, University of Hull, Hull, UK, September 1994, pp.18-21.

[21] Soloman, M.B." What's wrong with multimedia in higher education?". Technological Horizons in Education Journal. 21(7), 1994, pp. 81-83.

[22] Tennyson, R.D. et al. "Employment of system dynamics in modelling of instructional design (ISD)". In: R.D. Tennyson and A.E. Barron, Editors: *Automating instructional design: computer-based development and delivery tools*. Berlin: Springer-Verlag, 1995, pp. 603-609.

[23] Marshall, I. M., W. B. Samson, P. I. Dugard and W. A. Scott. "Predicting the Development Effort of Multimedia Courseware", Information & Software Technology, Vol 36, No. 5, 1994, pp. 251-258.

[24] Marshall , I. M. "Evaluating Courseware Development Effort Estimation Measures and Models". Ph.D. thesis. University of Abertay Dundee, 7 November 1996.

[25] Naur P., Randell B. (eds*) Software Engineering: A Report on a Conference sponsored by the NATO Sciece Committee*. NATO Scientific Affairs Division, 1969.

[26] Gibbs W.W. "Software's Chronic Crisis". Scientific American, Sep. 1994, pp 86-87.

[27] Laudon K.C., Laudon, J.P. *Management Information Systems. New Approaches to Organization*. Ed. Prentice Hall, 1999.

[28] Ibrahim , B. "Software Engin eering Techniques for CAL". Education and Computing 5(4), 1989, pp.215-222.

[29] diSessa, A.A. "A Principled Design for an Integrated Computational Environment". Human-computer interaction, 1, 1985, pp.1-47.

[30] Hurley, S. and Stephens, N.M. "Courseware in High Performance Computing". Proceedings of the International Conference on Parallel Computing for Undergraduates, 1994.

[31] Hurley, S, Marshall, A.D., McIntosh-Smith, S.N., Stevens, N.M. "Courseware for Parallel Computing using Mosaic and the World Wide Web", Proceedings of the 2nd International WWW Conference '94, 1, 1994, pp 499-508.

[32] Marshall, A.D. , Hurley, S., McIntosh-Smith, S.N., Martin, R.R. and Stevens, N.M. "Novel Uses of Computers for Teaching". AXIS: The UCISA Journal of Academic Computing and Information Systems, 1(3), 1994, pp 30-41.

[33] Pressman R.S. *Software Engineering: A Practitioner 's Approach*. McGraw-Hill Book Company (New York), third edition, 1992.

[34] McIlroy M.D." Mass Produced Software Components". In *Software Engineering*, P. Naur and B. Randell, editors,. NATO Science Committee, January 1969.

[35] Szyperski C. *Component Software - Beyond Object-Oriented Programming*. ACM Press/Addison-Wesley, 1998.

[36] Sametinger J. *Software Engineering with Reusable Components*, Springer, New York, 1997.

[37] Gagné, R., Briggs, L., Walter, W. *Principles of Instructional Design*. Harcourt, Brace, Jovanovich, New York, 4th ed., 1992.

[38] Reigeluth, C. (1983). Instructional Design: What Is It and Why Is It? In Reigeluth, ed., Instructional-Design Theories and Models: An Overview of their Current Status, p.7. Hillsdale, NJ: Lawrence Erlbaum Associates.

[39] Galvis Panqueva, A.H. *Ingeniería del Software Educativo*. Ediciones Uniandes, Santafé de Bogotá, Colombia, 1992.

[40] Harding R.D et al. "A consortium approach to courseware design in mathematics". Computers & Education, 26, 1996, pp.171-178.

[41] Neilson I., Thomas R. "Designing educational software as a re-usable resource". J. of Computer Assisted Learning, 12, 1996, pp. 114-126.

[42] Marshall D."Developing Interactive Courseware on the World Wide Web". Innovations in Education and Training International 36(1), 1999, pp. 34-43

[43] Rogerson D. *Inside COM: Microsoft's Component Object Model*. Microsoft Press, 1997.

[44] Sun MicroSystems. *Enterprise JavaBeans Specification*. August, 1999.

[45] Object Management Group . *The Common Object Request Broker: Architecture and Specification*, July 1996.

[46] Wiley, D. A. "Learning object design and sequencing theory". Unpublished doctoral dissertation, Brigham Young University, 2000. http://davidwiley.com/papers/dissertation/dissertation.pdf

[47] Sommerville , I. *Software Engineering*.. Addison-Wesley, 6th edition, 2002.

[48] Schafer , W. , Prieto-Díaz , R. , Matsumoto , M.*Software reusability*. Ellis Horwood Workshop series, UK, 1994.