Un Modelo Evolutivo para el Desarrollo de Cursos Apoyados por Tecnología, Orientado a la Mejora Permanente del Proceso de Enseñanza-Aprendizaje

Sergio F. Ochoa D., David A. Fuller P.
Pontificia Universidad Católica de Chile
Escuela de Ingeniería. DCC. Santiago, Chile
{sochoa, dfuller}@ing.puc.cl

Resumen

Últimamente, las instituciones educativas han comenzado a ver al alumno como un cliente, al cual le brindan un servicio (educación). Además, debido a la globalización, tanto los clientes como los prestadores del servicio están disponibles desde casi cualquier punto del planeta, a través de Internet. Esto ha sumido a las instituciones educativas en una búsqueda desesperada por tratar de mejorar la calidad de su servicio (educación), con el objetivo de ganar mercado. Estas instituciones han abordado el desafío, fundamentalmente a través del uso de nuevas tecnologías. Lamentablemente, no hay un camino de referencia que indique cómo avanzar hacia el mejoramiento del servicio, utilizando este tipo de herramientas. El presente trabajo propone un modelo evolutivo para el desarrollo de cursos apoyados por tecnología, el cual está orientado a la mejora permanente del proceso de enseñanza-aprendizaje. Este modelo establece un camino de referencia para mejorar la calidad de la educación apoyada por tecnología (servicio); en principio, a nivel universitario y en el ámbito de la ingeniería.

Palabras Claves: Modelo de Desarrollo de Coursewares, Educación Apoyada por Tecnología, Courseware.

1. Introducción

En los últimos años, principalmente a partir del trabajo de W. E. Deming **[Dem86]**, el concepto de *calidad* se ha ido masificando. El éxito alcanzado por la industria japonesa demostró que el camino hacia el mejoramiento de la calidad de procesos y productos, no sólo era rentable, sino que además era necesario **[Wak99]**. Este concepto ha ido extendiéndose a otras áreas, entre las cuales se encuentra la *educación* **[Wak99]**. Para mejorar la calidad de la educación, las instituciones han optado principalmente por el uso de herramientas de Courseware **[Häm96, Sch98]**, y por el rediseño de las estrategias de enseñanza-aprendizaje, para aprovechar los aportes de la tecnología **[McN99]**. Lamentablemente, este no es el camino trazado por la industria para el mejoramiento de procesos y productos. Recomendaciones ampliamente aceptadas como CMM (Capability Maturity Model) **[Pau93]** y TQM (Total Quality Management) **[DoD89, Ban00]**, indican que el mejoramiento de la calidad es un proceso que comienza con la estabilización de dicho proceso. Un *proceso estable* es aquel que es capaz de *repetir*, en cada aplicación, los resultados de situaciones anteriores. Entonces la pregunta es: ¿cómo lograr un proceso estable?.

Los procesos en general se ven afectados por tres componentes: *Personas*, *Herramientas* y *Modelos* **[Hum95, Jac99]**. Las *personas* llevan adelante el proceso, utilizando las *herramientas* adecuadas, y siguiendo las recomendaciones de los *modelos* de desarrollo para dicho proceso. Esta regla también incluye al proceso de enseñanza-aprendizaje.

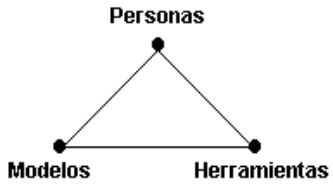

Figura 1. Elementos que afectan a un proceso

Respondiendo a la pregunta anterior, para lograr un *proceso estable* es necesario mantener un rendimiento estable por parte de cada uno de estos componentes (personas, herramientas, y modelos). Esto se puede esperar de las herramientas y de los modelos, pero no de las personas. En el caso del proceso de enseñanza-aprendizaje, el problema es mayor aún, ya que la calidad de este proceso depende fundamentalmente del profesor a cargo (Personas, Fig. 1) **[Rup99]**.

Esto es lo que se conoce como un *proceso artesanal*. Y como cualquier proceso artesanal, no es independiente de quien lo ejecuta, ni es reproducible, ni se pueden esperar resultados mínimos **[Hum95]**. Además, la capacidad de evolución de los procesos artesanales es incierta **[Hum95]**, y esto obstaculiza cualquier intento por mantener o mejorar la calidad del proceso y/o del producto. En otras palabras, el futuro de una institución o de un sistema educativo es incierto, debido a que no se puede planificar un camino de crecimiento, sobre la base de un proceso artesanal.

Poco se ha hecho para resolver este problema. Las certificaciones ISO 9000 y ABET (Accreditation Board for Engineering and Technology), han ayudado a medir la calidad de los procesos involucrados, pero no a mejorarla. Hay un proyecto llamado EONT (An Experiment in ODL Using New Technologies), que tiene por objetivo experimentar con el proceso de construcción y dictado de cursos a distancia **[Ret97]**. Para ello, siete importantes universidades de Europa han trabajado en la creación de un modelo que propone una metodología de desarrollo en cascada, para la etapa de construcción de un curso; y una serie de recomendaciones para la etapa de producción. Lamentablemente, el proyecto no contempla, entre otras cosas, la evolución del proceso (mejoramiento).

El resto de las iniciativas involucran el uso de tecnología **[Häm96, Sch98]**, y en muy pocos casos, la redefinición de las estrategias de enseñanza-aprendizaje **[McN99]**. Esto tampoco sirve, si los intentos de solución no forman parte de un modelo que entregue resultados estables, e información útil para la evolución del proceso de enseñanza-aprendizaje.

Fuera del área de educación, hay recomendaciones muy precisas para solucionar el problema de la artesanalidad de un proceso, como por ejemplo CMM (Capability Maturity Model) **[Pau93]** y TQM (Total Quality Management) **[DoD89, Ban00]**, pero ambas son muy complejas para ser aplicadas en su totalidad en el área de Educación. Es por eso que el modelo evolutivo propuesto rescata los componentes más relevantes de estas dos recomendaciones, tratando de simplificar al máximo el esfuerzo necesario para su aplicación. Este modelo tiene como objetivo *estabilizar* el proceso de enseñanza-aprendizaje en cursos de nivel universitario (pregrado y postgrado), apoyados por tecnología, en el área de ingeniería. Además como complemento necesario para la evolución del proceso (según CMM y TQM), el modelo ofrece un camino para comenzar con el *mejoramiento continuo* del mismo. Como lo demuestra la historia, este mejoramiento continuo del proceso lleva a un *mejoramiento seguro* del producto **[Dem86, Ban00]**.

Este trabajo abarca las siguientes áreas: CSCL (Computer-Supported Collaborative Learning), CSCW (Computer-Supported Collaborative Work), Ingeniería de Software, y Educación Apoyada por Tecnología. Su ámbito es el proceso de enseñanza-aprendizaje en ingeniería, realizado mayormente en

forma presencial y con apoyo tecnológico (curso en Web), en pregrado y postgrado. A futuro, se analizarán posibles extensiones.

1.1. Introducción a la Solución Propuesta

Este trabajo pretende: "*Transformar el **proceso artesanal** de enseñanza-aprendizaje, en un **proceso evolutivo** y **controlado**, orientado a la mejora permanente de dicho proceso*". Esto no significa que el modelo propuesto mejorará la calidad del proceso de enseñanza-aprendizaje por el solo hecho de aplicarlo. Sino que los resultados obtenidos por aplicar el modelo, estarán dentro de un cierto rango de valores, independientemente del profesor a cargo del curso. El mejoramiento de la calidad de la educación se dará, cuando se logre desplazar hacia arriba el rango de valores esperados por la aplicación del modelo. Este rango de valores podrá desplazarse, como consecuencia de la reingeniería de un courseware, hecha en base a la retroalimentación arrojada por el modelo. De esa manera el proceso de enseñanza-aprendizaje, comenzará a ser evolutivo, controlado, y orientado a la mejora permanente.

Para lo antes mencionado, se utilizarán los patrones que guiaron la evolución de los servicios de atención al cliente apoyados por computador, fundamentalmente porque ambos procesos tienen la misma estructura **[Och00]**. Esto se debe a que el proceso de enseñanza-aprendizaje puede considerarse como un *servicio* que una institución educativa (prestador del servicio) ofrece a sus *clientes* (alumnos) **[Och99a, Tsi99]**.

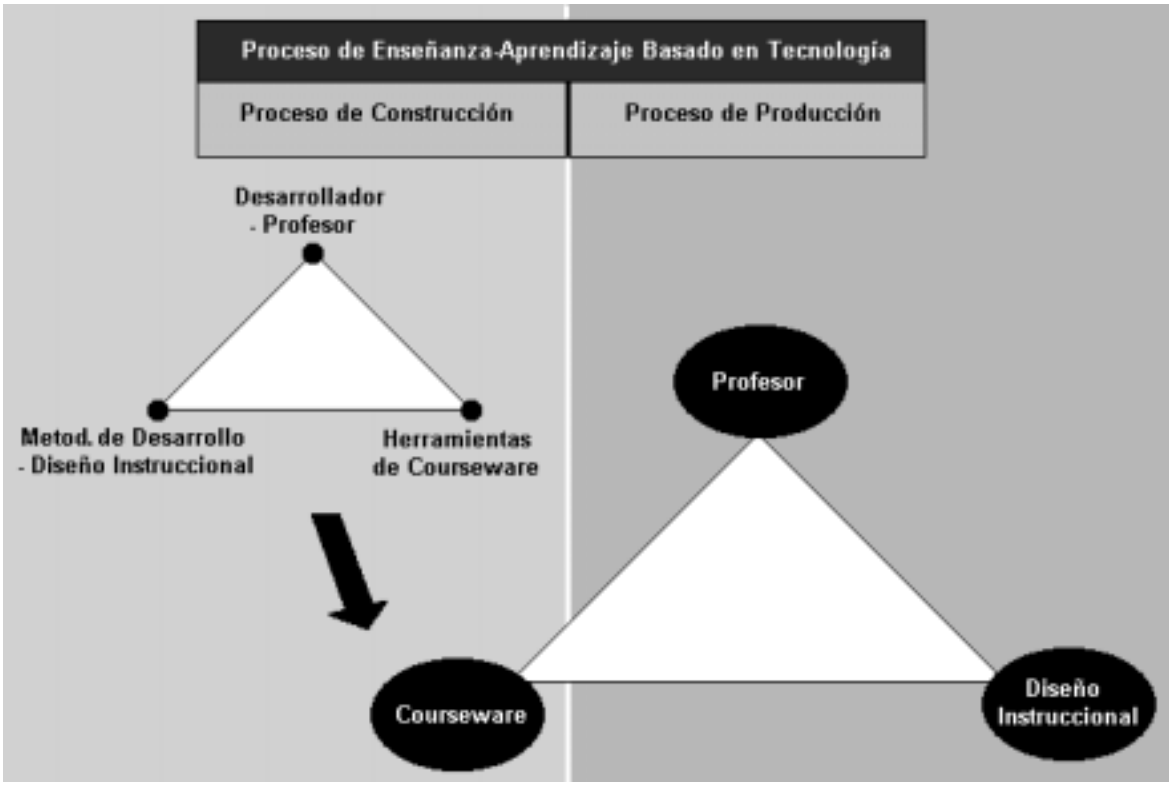

Figura 2. Estructura del proceso de enseñanza aprendizaje

Ambos procesos están compuestos por un proceso llamado de *Construcción* y otro de *Producción* (Fig. 2), y están afectados por los tres elementos antes mencionados (Fig. 1). El ***proceso de construcción*** tiene como objetivo diseñar y construir el sistema computacional de apoyo al servicio de información/educación, e involucra a *Personas* (Equipo de Desarrollo, Usuario del Sistema / Asistente de un Curso, Profesor), *Modelos* (de Desarrollo de Software, Propios del Dominio del Problema / de Desarrollo de Coursewares, Instruccionales), y *Herramientas* (de Desarrollo de Software, Propias del Dominio del Problema / de Desarrollo de Coursewares, Instruccionales).

El ***proceso de producción*** tiene como objetivo brindar el servicio al cliente, utilizando el sistema computacional construido en la etapa anterior, y también involucra *Personas* (de Atención al Cliente, Cliente / Profesor, Alumno), *Modelos* (Técnicas de Atención al Cliente, Técnicas de Marketing / Instruccionales), y *Herramientas* (Sistemas de Información / Coursewares).

El proceso de atención al cliente (que también en un principio era artesanal), basó su evolución en la incorporación de la inteligencia (teorías propias del dominio, experiencia, etc.) proveniente de los *modelos* y las *personas*, a la *herramienta* de apoyo al proceso (sistema de información). De esa manera, redujo la responsabilidad de las personas sobre el proceso, aumentando la responsabilidad de las herramientas. Y debido a que las herramientas son más predecibles que las personas, se aumenta la predictibilidad de la *calidad del servicio* entregado al cliente.

De la misma manera, para reducir el grado de artesanalidad del proceso de enseñanza-aprendizaje, el modelo propuesto busca incorporar en la herramienta de apoyo al curso (courseware), gran parte de la inteligencia que hoy reside en los profesores (experiencia) y en los modelos de educación (diseño instruccional). De esa forma, el problema se reduce a: ¿Cómo desarrollar un courseware de calidad predecible, que sea capaz de incorporar el conocimiento propio del dominio?. La respuesta a esta pregunta se puede encontrar siguiendo los 50 años de evolución de la ingeniería de software desde el modelo Code-and-Fix **[Boe88]** hasta el RUP (Rational Unified Process) **[Jac99]**.

2. Descripción de la Propuesta

El modelo evolutivo propuesto (Fig. 3) aprovecha los últimos 50 años de experiencia de la ingeniería de software. Esta disciplina impuso un modelo de desarrollo de sistemas de información usando tecnología de orientación a objetos, que consta de una *metodología de desarrollo de software* basada en prototipos (evolutiva o incremental), un *lenguaje de modelamiento de sistemas* (Booch Notation, OMT, o UML) y *componentes de software* (ActiveX, JavaBeans, Corba Beans, OpenDoc, o Clases con alta cohesión)**[Orf98]** que por lo general están organizados en un framework **[Joh97, Rob97]**. Este es el modelo de desarrollo más popular en la actualidad, debido a las ventajas mostradas con respecto a la predictibilidad de los resultados del proceso de desarrollo. Como ejemplo de la adhesión a este modelo podemos mencionar al RUP (Rational Software Process) también conocido como "The Unified Software Development Process" **[Jac99]**.

Aprovechando la experiencia de la ingeniería de software, el modelo evolutivo propuesto, también consta de los tres elementos fundamentales del modelo anterior: una metodología de desarrollo, un lenguaje de modelamiento, y un framework de componentes de software, aunque éstos han sido especializados para el dominio de la educación apoyada por tecnología. En este escenario, la metodología es la encargada de guiar el proceso de desarrollo de un courseware, utilizando el lenguaje

de modelamiento para la etapa de diseño, y el framework de componentes para la etapa de implementación.

Hay seis ítems que deben ser manejados por cualquier modelo que pretenda atacar el problema de la artesanalidad del proceso de desarrollo de software (representada por la "crisis del software") **[Jac99]**. Estos ítems son: Desarrollo Evolutivo, Uso de Arquitecturas de Componentes, Administración de Requisitos, Modelamiento Visual, Control de Cambios, y Verificación de la Calidad. El modelo propuesto maneja completamente los 4 componentes primeros, y parcialmente los otros dos.

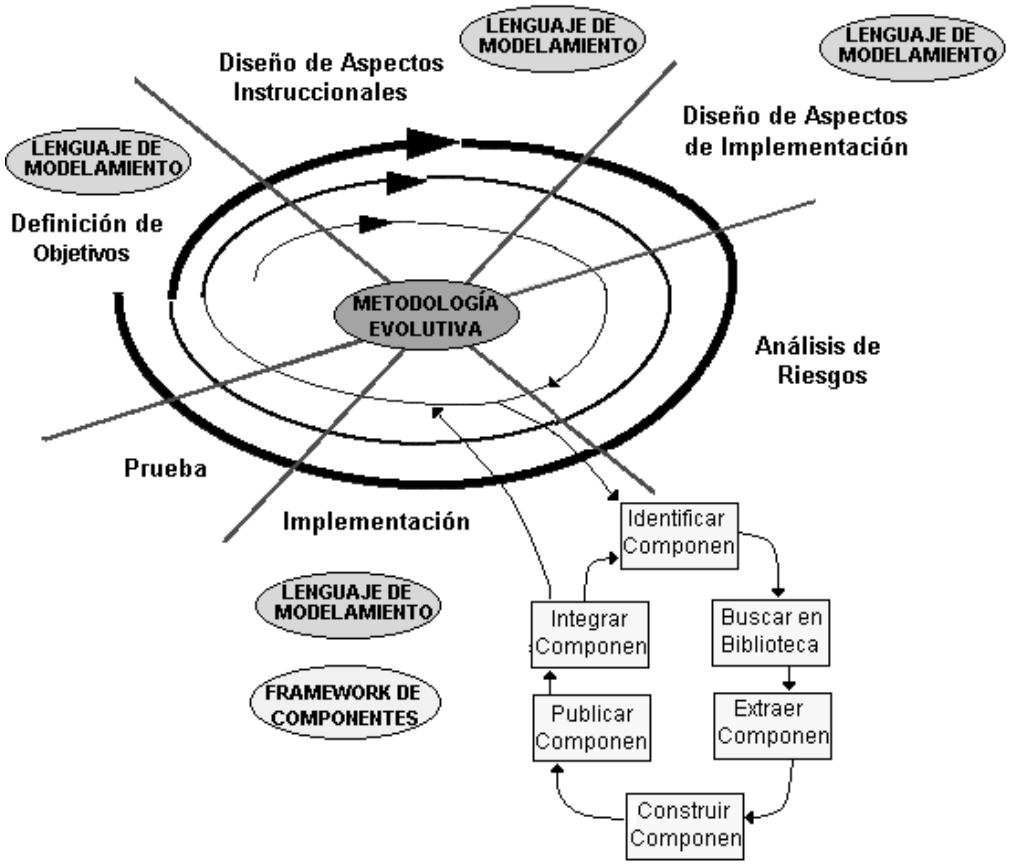

Figura 3. Esquema General del Modelo Propuesto

Para su aplicación, el modelo propuesto recibe como entrada los contenidos mínimos (si los hay) y los objetivos de un curso, y produce un ambiente educativo basado en Web (courseware), de apoyo al curso. Allí se encuentran publicados los contenidos curriculares, y las actividades diseñadas para alcanzar los objetivos de dicho curso. Además, este ambiente generado tiene capacidades de comunicación y colaboración tanto sincrónica como asincrónica entre los usuarios conectados a él.

Los cursos creados utilizando el modelo, apoyan al profesor en la tarea de llevar adelante el proceso de enseñanza-aprendizaje. Éstos son construidos a través del ensamble de módulos predefinidos (componentes de software) que almacenan mecanismos de colaboración entre usuarios, conocimiento instruccional, experiencia docente, capacidades de reusabilidad e interoperabilidad, y una interfaz

definida. De esa manera se puede abstraer al implementador de cursos, de los detalles de implementación, permitiéndole utilizar/reutilizar fácilmente tanto la inteligencia como los datos almacenados en los componentes. El modelo propuesto brinda un marco de referencia inicial para llevar adelante el proceso de enseñanza-aprendizaje. Este marco de referencia debe ser adaptado a las particularidades de cada institución.

A continuación se describen los tres componentes principales del modelo: la metodología evolutiva (sección 2.1), el lenguaje de modelamiento (sección 2.2), y el framework de componentes (sección 2.3).

2.1. Metodología Evolutiva

En la figura anterior (Fig. 3) se puede apreciar el diseño global de la metodología de desarrollo propuesta, que es una adaptación al dominio educativo, de la metodología de Nierstrasz **[Nie92]**. Al igual que su precursora, la metodología propuesta está basada en objetivos, y en su diseño preliminar **[Och00]** consta de tres iteraciones (o fases): a nivel de *objetivos fundamentales*, a nivel de *objetivos específicos*, a nivel de *subobjetivos*. Los distintos niveles de los objetivos corresponden a los tres niveles de abstracción que han sido definidos para atacar el problema de diseño de un curso. En cada fase se diseña una parte distinta, pero los niveles inferiores tienen que respetar las restricciones impuestas por los niveles superiores. Además cada fase consta de seis etapas, cuya función depende de los objetivos de la fase. Las etapas a desarrollar en cada iteración (o fase) son:

Definición de Objetivos: Como su nombre lo indica, el objetivo de esta etapa es definir claramente y sin ambigüedades, los objetivos a alcanzar (requisitos a cumplir) durante la ejecución de la fase. Si la fase actual es una fase intermedia (no es la primera fase), los objetivos de ésta deberán respetar las restricciones impuestas por las fases anteriores.

Diseño de Aspectos Instruccionales: Recibe como entrada los objetivos (requisitos) de la fase, y produce un diseño de los aspectos instruccionales del producto (objetivos) de la misma. Entre los elementos que se utilizan para generar este diseño están los contenidos, las actividades, y las estrategias de enseñanza-aprendizaje.

Diseño de Aspectos de Implementación: Durante esta etapa se construye uno o más diseños del producto de la fase. Éstos deben estar lo suficientemente especificados como para ser validados en la siguiente etapa (análisis de riesgos), y posteriormente implementados en la etapa de implementación. Además, deben respetar lo especificado en las etapas anteriores, y deben carecer de ambigüedades.

Análisis de Riesgos: En esta etapa se someten a prueba los diseños realizados en las dos etapas anteriores. Para ello se valida en forma estática el cumplimiento de los objetivos de la fase, y se identifican posibles fuentes de problemas en los diseños realizados. También se validan los recursos tecnológicos a utilizar para implementar los diseños.

Implementación: Durante esta etapa se implementa el o los productos de la fase, en base a los diseños desarrollados en las etapas anteriores. Para ello, se pueden utilizar los componentes de software instanciados, o sin instanciar, que formen parte de la biblioteca de componentes

(framework). Además, si un componente requerido para la implementación no forma parte del framework, éste puede ser construido utilizando componentes menores del framework, o bien puede ser desarrollado en forma totalmente aislada. Cualquiera sea el método elegido, el componente deberá incorporarse al framework para poder ser utilizado/reutilizado, en una implementación.

Prueba: La función de esta etapa es validar que los objetivos de la fase se han cumplido. Si no es así, la metodología obliga a realizar una nueva iteración, partiendo con la revisión y/o redefinición de los objetivos iniciales. Las características de la prueba dependerá de la fase que se esté desarrollando, y puede involucrar a profesores, ayudantes, y alumnos. La etapa de prueba, al igual que la de análisis de riesgos, tiene además la misión de obtener retroalimentación para mejorar el proceso.

La metodología propuesta utiliza el lenguaje de modelamiento (sección 2.2) para diseñar los distintos aspectos del curso, y el framework de componentes (sección 2.3) para la implementación de los diseños. Una característica importante de la metodología es que debido a que está estratificada, permite el trabajo en paralelo entre las fases.

2.2. Lenguaje de Modelamiento

Los lenguajes de modelamiento son generalmente visuales, y se utilizan en áreas como: representación de conocimiento **[Jon99, Her00]**, ingeniería de software **[OMG99]**, y educación **[Lam84, Don97, Nov98, Bal99]**, entre otras. Estos lenguajes utilizan *nodos* y *arcos* para implementar sus modelos, que son organizados a través de distintos tipos de grafos. Según el lenguaje utilizado, un *nodo* puede ser llamado: clase, concepto, vértice, etc. Y un arco puede ser llamado: lado, vínculo, relación, etc. De todos modos lo que quieren representar los nodos y los arcos de estos lenguajes, es aproximadamente lo mismo.

Este trabajo propone un lenguaje de modelamiento gráfico, orientado a la educación, para permitir el modelamiento de los distintos aspectos de un curso apoyado por tecnología. El lenguaje está influenciado por la teoría del aprendizaje significativo **[Aus83, Nov98]**, por lo tanto el modelamiento de los conocimientos se realiza a través de distintos símbolos con una semántica única y predefinida. La apariencia de un modelo realizado con este lenguaje, es una mezcla entre Mapas Conceptuales **[Nov98]** y los Diagramas Estáticos de Clases de **UML [OMG99].**

Este lenguaje utiliza dos tipos de diagramas para modelar un curso: *diagramas de contenidos*, y *diagramas de objetivos*. El diagrama de contenidos, representa el árbol de conocimientos a entregar, y las actividades previstas para alcanzar los objetivos de un curso. En cambio el diagrama de objetivos relaciona los distintos objetivos de un curso, con los contenidos y actividades previstas.
El lenguaje permite el modelamiento estático de un curso, utilizando los siguientes cuatro componentes:

Contenidos: Son los nodos del diagrama de contenidos, y almacenan material didáctico de distintos tipos. Dentro de los tipos de contenidos válidos están los capítulos, secciones, ejemplos, y problemas, entre otros.

Actividades: También son nodos del diagrama de contenidos, y se representan gráficamente con un símbolo distinto al de los contenidos. Entre los tipos de actividades previstas están los exámenes, tareas grupales, investigaciones grupales, lluvia de ideas, y discusiones.
Objetivos: Representan los nodos del diagrama de objetivos, y están vinculados a los contenidos y a las actividades. De esa manera es más fácil constatar si las actividades y contenidos previstos, son suficientes para alcanzar los distintos objetivos del curso. Entre los distintos tipos de objetivos están los objetivos fundamentales, específicos, y subobjetivos **[MEC98]**.
Relaciones: Vinculan a los distintos tipos de nodos (contenidos, actividades, y objetivos) permitidos en el lenguaje. Estas relaciones pueden ser de distinto tipo (relaciones simples, composición, agregación, etc), para lo cual tienen distinta representación gráfica.

En el área de educación, se han desarrollado varios lenguajes de modelamiento gráfico, como por ejemplo: mapas conceptuales **[Nov98]**, redes semánticas **[Lam84]**, redes didácticas **[Bal99]**, y mapas espaciales **[Don97]**. Estos lenguajes no son muy útiles para modelar cursos, debido a que no han sido concebidos para esa tarea. En cambio, el lenguaje propuesto está específicamente diseñado para esa tarea. Entre las ventajas que éste aporta, está el uso de la *estructura* (capítulos, secciones, etc.), la cual es bien conocida y manejada por los educadores, y además ésta se ajusta a mayoría de las reglamentaciones educativas actuales. Otro punto importante a favor del lenguaje propuesto es que su notación es más rica, y con una semántica es más fuerte, lo cual le brinda un mayor poder de expresión. Finalmente, permite una validación más completa de los diseños, debido a que los modelos combinan contenidos, actividades, y objetivos a cumplir.

2.3. Framework de Componentes de Software

El framework propuesto implementa todos los elementos del lenguaje de modelamiento, de manera que exista un vínculo directo entre el modelo diseñado y el implementado. Esto facilita la ingeniería, el reuso de componentes, y la evolución de un curso. El framework también puede almacenar en cada uno de sus componentes, el trabajo y la experiencia de expertos en cada una de las áreas involucradas en el producto. Además, permite a los implementadores de cursos (profesores o ayudantes) abstraerse de los detalles de implementación, y construir un curso reusando y adaptando un conjunto de módulos predefinidos (componentes). Esto generalmente reduce el esfuerzo (tiempos y costos) asociado a esta tarea, y aumenta la calidad del producto final **[Pre97, Boe99]**.

Los *componentes* son piezas de software identificables, con límites bien definidos (alta cohesión), y que sirven para la construcción de software a través del ensamble de éstos. Puede concebírselos como un *bloque Lego* **[Orf98]**. Estos componentes generalmente están organizados en un *framework*, que es capaz de agrupar a los componentes relacionados, y de agregarle semántica a estas relaciones **[Rob97]**.

El framework propuesto está inserto en una arquitectura cliente/servidor estratificada, que separa en tres capas, la problemática asociada a la implementación de un courseware. Esto le brinda ventajas ya conocidas, propias del uso de este tipo de arquitectura.

Figura 4. Arquitectura en la que está inserto el Framework

Como se puede ver en la *figura 4*, los componentes del framework son el vínculo entre un courseware, y los mecanismos necesarios para la comunicación, la coordinación, y para compartir objetos. Por estar apoyados sobre una plataforma de objetos distribuidos, los componentes del framework propuesto incorporan mecanismos de colaboración (sincrónica / asincrónica) apropiados, para que los usuarios puedan colaborar en forma fácil. Además, estos componentes pueden integrarse con cualquier otro componente que adhiera a la misma especificación, promoviendo así el reuso tanto de diseños como de conocimientos.

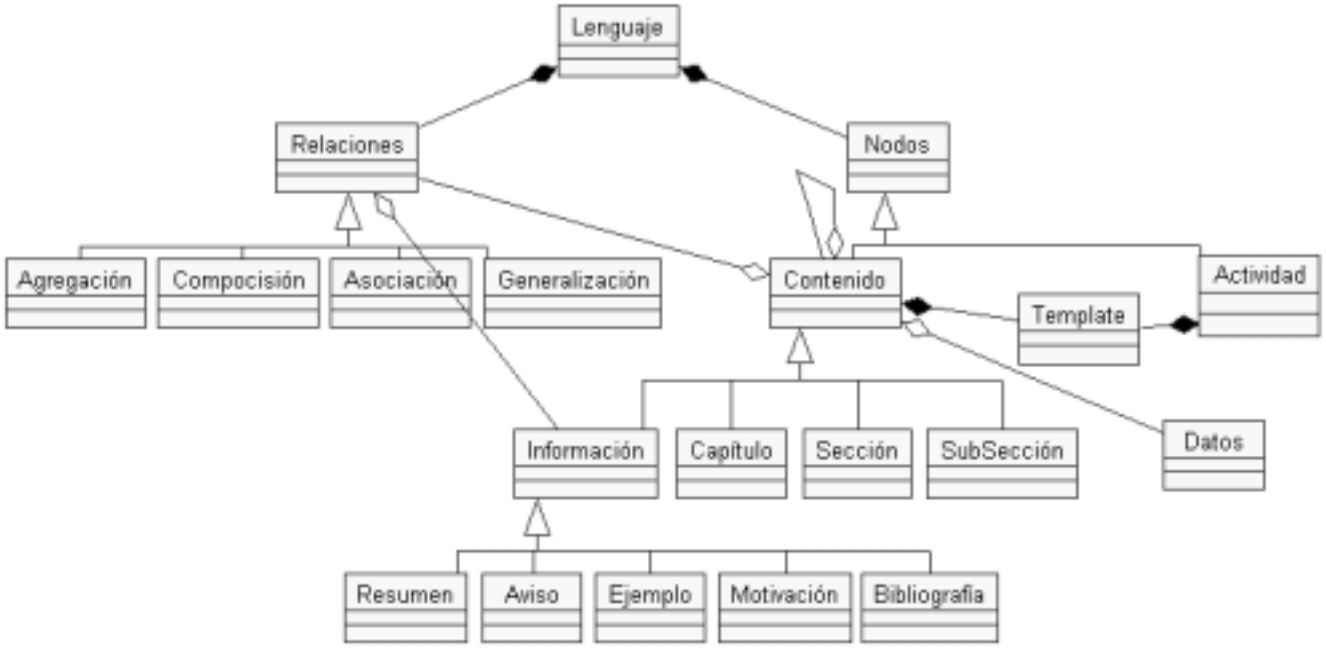

Figura 5. Parte del Framework de Componentes

Durante la segunda mitad de los ´90 se desarrollaron una gran cantidad de frameworks en el área de educación, entre los que figuran: CAIRO **[Hus95]**, CMap **[Kre97]**, COCHI **[Lic00]**, ODLE **[Müh97]**, y TOP **[Gue99]**. Éstos tratan de aportar soluciones, a través de un reuso limitado de código y diseños. La limitación se debe a que ninguno de ellos está compuesto por componentes de software que adhieran a especificaciones abiertas, las cuales aseguran la capacidad de reuso e interoperabilidad entre sus componentes (por ej. JavaBeans, ActiveX, o Corba Beans). Hay también herramientas de courseware **[Lan00]** que están basadas en frameworks, pero debido a que éstas han pasado a ser comerciales, la información sobre esos frameworks ya no está disponible. Un ejemplo de esto es Virtual-U **[Har99]**.

También se están desarrollando componentes educativos que adhieren a estándares abiertos **[Ros99]**, pero éstos apuntan al desarrollo de aplicaciones educativas en temas específicos (matemáticas, física, etc.), y además no cuentan con un framework que los integre y los ordene.

3. Resultados Obtenidos, y Esperados

El modelo propuesto se encuentra en la etapa final de su diseño, aunque varios de sus componentes ya han sido probados utilizando la herramienta ICESE **[Och99b]**. El conocimiento adquirido debido a la

experimentación con esta herramienta, a alentado el desarrollo del modelo propuesto, y ha servido como un medio para probar algunas de las hipótesis de este trabajo, antes de que el framework esté totalmente implementado.

Entre las cosas que se han probado está la metodología de desarrollo (sección 2.1) **[Och00]**, la cual ha mostrado buenos resultados en cuanto a la calidad del producto final, manteniéndose el esfuerzo necesario para la implementación. Los resultados obtenidos en dos aplicaciones consecutivas del modelo, han sido muy similares, aunque no se cambió al profesor a cargo del curso. Éste será el próximo paso a seguir.

Además se ha probado la portabilidad y reusabilidad de algunos de los componentes, especialmente capítulos, secciones y subsecciones. Los resultados obtenidos fueron muy buenos, y de magnitud similar a lo planificado. Por otra parte, en este momento se está probando el lenguaje de modelamiento, pero aún no se ha concluido con la etapa de experimentación, por lo tanto no hay valores de referencia al respecto.

4. Conclusiones

Independientemente de las ventajas que significa contar con un ***modelo evolutivo de desarrollo*** de cursos apoyados por tecnología, un ***framework*** de componentes de software orientado a la educación, un ***lenguaje de modelamiento*** de cursos, y una ***metodología evolutiva*** para el uso, reuso e instanciación de los componentes del framework; el gran aporte de este trabajo está en la transformación planteada. Transformar el proceso artesanal de enseñanza-aprendizaje, en un proceso predecible es un importante desafío, que podría cambiar el desarrollo futuro de este proceso. Si bien por ahora el problema a atacar está limitado al área de ingeniería, futuras extensiones podrían permitir una aplicación masiva del modelo en la sociedad. Esto le dará a las instituciones educativas y de gobierno, una base más segura para la planificación y ejecución de sus proyectos. El modelo propuesto además podría ayudar a las instituciones educativas a certificar la calidad de sus procesos educativos (ISO 9000, ABET), y a entrar al camino de la *mejora continua de la calidad* tanto de procesos, como de productos.

Este trabajo propone un enfoque inédito en el área de educación asistida por computador, y si bien los resultados no pueden conocerse sino hasta realizar un experimento científicamente válido, los resultados obtenidos en otras áreas hacen esperar un aporte importante de esta propuesta.

Reconocimientos

Este trabajo ha sido parcialmente financiado por el Fondo Nacional de Ciencia y Tecnología de Chile (FONDECYT), Proyecto Nº 198-0960.

Referencias

[Aus83] D. Ausubel, J. Novak, H. Hanesian. *Psicología Educativa: Un Punto de Vista Cognoscitivo*. (2º Ed.). Trillas, México. 1983 .

[Bal99] N. Baloian, J. Pino, U. Hope. *Intelligent Navigation Support for Lecturing in an Electronic Classroom.* Artificial Intelligence in Education. S Lajoie and M. Vivet (Eds.). pp. 606-610. IOS Press. 1999.

[Ban00] J. Bank. *The Essence of TQM*. (2º Ed.). Prentice Hall. 2000.

[Boe88] B. Boehm. *A Spiral Model for Software Development and Enhacement*. IEEE Computer, Vol. 21, Nº 5, pp. 61-72. May. 1988.

[Boe99] B. Boehm. *Managing Software Productivity and Reuse*. IEEE Computer, Vol. 32, Nº 9, pp. 111-113. Sep. 1999.

[Dem86] W. E. Deming. *Out of the Crisis*. Massachusetts Institute of Technology, Center for Advanced Engineering Study. Cambridge, Mass. USA. 1986.

[DoD89] DoD 5000.51-G. *Guidelines for Total Quality Management.* Office of the Deputy Assistant Secretary of Defence for Total Quality Management. Pentagon, Washington, D.C., USA. 1989.

[Don97] S. McDonald, R. Stevenson. *The Effects of a Spatial Map and a Conceptual Map on Navigation and Learning Hypertext*. Proc. of ED-MEDIA ´97. Calgary. Alberta. Canada. Jun.14-19. 1997.

[Gue99] L. Guerrero, D. Fuller. *A Web-Based OO Platform for the Development of Multimedia Collaborative Applications*. Decision Support Systems Journal, Vol. 27, No. 3, pp. 257-270. Nov. 1999.

[Häm96] M. Hämäläinen, A. Whinston, A. Vishik. *Electronic Markets for Learning: Education Brokerages on the Internet*. CACM, Vol. 39, Nº 6, pp. 51-58. Jun. 1996.

[Har99] L. Harasim. *A Framework for OnLine Learning: The Virtual-U*. IEEE Computer, Vol. 32, Nº 9, pp. 44-49. Sep. 1999.

[Her00] O. Herrera, S. Ochoa, V. Rodríguez, D. Fuller. *Shaknoma: A Collaborative Tool for Shared Knowledge Management*. Submitted to revision to ED-MEDIA 2000. Jun. 26 – Jul. 1. 2000. Montreal, Canada.

[Hum95] W. Humphrey. *A Discipline for Software Engineering*. Addison-Wesley. 1995.

[Hus95] K. Hussein, F. Pena-Mora and R. Sriram. *CAIRO: A System for Faciltating Communication in a Distributed Collaborative Engineering Environment*. In Proc. 4º IEEE Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. pp.174-183. Apr. 1995.

[Jac99] I. Jacobson, G. Booch, J. Rumbaugh. *The Unified Software Development Process*. Addison Wesley. 1999.

[Joh97] R. Johnson. *Frameworks = (Components + Patterns)*. CACM, Vol. 40, Nº 10, pp. 39-42. Oct. 1997.

[Jon99] C. Jonker, R. Kremer, P. Leeuwen, D. Pan, J Treur. *Visual and Textual Knowledge Representation in DESIRE*. In: I. Imam, Y. Kodratoff, A. El-Dessouki, and M. Ali (Eds.), Multiple Approaches to Intelligent Systems (Proc. of the IEA/AIE '99). Lecture Notes in AI, Vol. 1611, pp. 306-315. Springer Verlag, 1999.

[Kre97] R. Kremer. *Constraint Graphs: A Concept Map Meta-Languaje*. PhD. Thesis. Univ. of Calgary, Calgary, Alberta, Canada, 1997.

[Lam84] J. Lambiotte, D. Dansereau, D. Cross, S. Reynolds. *Multirelational Semantic Maps.* Educational Psychology Review. Vol. 1, Nº 4, pp. 331-367. 1984.

[Lan00] B. Landon. *OnLine Educational Delivery Applications: A Web Tool for Comparative Analysis*. Standing Commitee on Educational Technology; Center for Curriculum, Transfer and Technology. 2000. URL: http://www.ctt.bc.ca/landonline/

[Lic00] G. Licea, J. Favela. *An Extensible Platform for the Development of Synchronous Groupware*. Accepted for publication in Information & Software Technology, Elsevier Science B. V. 2000.

[McN99] C. McNaught. *Evaluation Strategies for Developing and Using Effective Computer-Facilited Learning Environments in Science and Engineering*. European Journal of Engineering Education. Vol. 24, Nº 1, pp.23-34. Mar. 1999.

[MEC98] Ministerio de Educación República de Chile. *Marco Curricular de la Educación Media*. Ministerio de Educación, Santiago, Chile. 1998.

[Müh97] M. Mühlenbrock, F. Tewissen, U. Hoppe. A Framework System for Intelligent Support in Open Distributed Learning Environments. Proc. of AIED ´97. Kobe. Japan. 1997.

[Nie92] O. Nierstrasz. *Component-Oriented Software Development*. CACM, Vol. 35, Nº 9, pp. 160-165. Sep. 1992.

[Nov98] J. Novak. *Learning, Creating, and Using Knowledge: Concept Maps As Facilitative Tools in Schools and Corporations*. Lawrence Erlbaum Assoc. 1998.

[Och99a] S. Ochoa, D. Fuller. *Una Metodología de Educación Basada en Componentes.* Memorias del CACIC ´99. Tandil. Argentina. 26-30 Oct. 1999.

[Och99b] S. Ochoa, D. Fuller. *Un Ambiente Colaborativo Integrado de Apoyo a la Educación (ICESE: Integrated Collaborative Environment to Support Education).* XXV Conferencia Latinoamericana de Informática (CLEI ´99). Asunción, Paraguay. 30 de Agosto a 3 de Septiembre de 1999.

[Och00] S. Ochoa, D. Fuller. *A Spiral Model for Technology Based Courses Development and Enhancement.* Accepted in INTERTECH 2000. Cincinati. Ohio. USA. Jun. 14-16. 2000.

[OMG99] OMG (Object Management Group). *Unified Modeling Language Specification, version 1.3*, Jun. 1999.

[Orf98] R. Orfali, D. Harkey. *Client/Server Programming with Java and CORBA*. John Wiley and Sons. 1998.

[Pau93] M. Paulk, B. Curtis, M. Chrissis, C. Weber. *Capability Maturity Model for Software, Versión 1.1*. Technical Report. CMU/SEI-93-TR-024. SEI. Carnegie Mellon University. 1993.

[Pre97] R. Pressman. *Ingeniería del Software, un Enfoque Práctico.* 4º Ed., McGraw Hill. 1997.

[Ros99] J. Roschelle, C. DiGiano, M. Koutlis, A. Repenning, N. Jackiw, D. Suthers. *Developing Educational Software Components*. IEEE Computer, Vol. 32, Nº 9, pp. 50-58. Sep. 1999.

[Ret97] S. Retalis, V. Makrakis, E. Skordalakis. *EONT Courseware Development Methodology.* Proc. of ED-MEDIA '97. Calgary. Alberta. Canada. Jun. 14-19. 1997.

[Rob97] D. Roberts, R. Johnson. *Patterns for Evolving Frameworks.* Proc. of PLOP 97 (Pattern Language Oriented Programing). Monticello. Illinois. USA. Sep. 2-5. 1997.

[Rup99] R. Ruprecht. *On a Threshold? A Critical Look at Quality Assessment*. European Journal of Engineering Education. Vol. 24, Nº 1, pp. 35-48. Mar. 1999.

[Sch98] R. Schank. *Horses for Courses*. CACM, Vol. 41, Nº 7, pp. 23-25. Jul. 1998.

[Tsi99] D. Tsichritzis. *Reingineering the University*. CACM, Vol. 42, Nº 6, pp. 93-100. Jun. 1999.

[Wak99] S. Waks, M. Frank. *Application of the Total Quality Management Approach Principles and the ISO 9000 Standards in Engineering Education*. European Journal of Engineering Education. Vol. 34, Nº 3, pp. 249-258. Sep. 1999.