PROYECTO CUPI2: UN ENFOQUE MULTIDIMENSIONAL FRENTE AL PROBLEMA DE ENSEÑAR Y APRENDER A PROGRAMAR[7]

Jorge A. Villalobos,[8] *Nadya A. Calderón*[9]

RESUMEN

Las dificultades en la enseñanza-aprendizaje de la programación han sido un problema recurrente en los últimos 20 años tanto en nuestro país como en el mundo entero. A lo largo del tiempo se han propuesto numerosas soluciones sin que ninguna haya resultado realmente efectiva. A los inconvenientes de motivación de los estudiantes se une la falta de un estudio a fondo de las habilidades que deben adquirir y en consecuencia, muchas veces, se reduce el programa de los cursos a un recorrido de estructuras sintácticas de un lenguaje de programación. Este documento plantea una nueva aproximación a la solución del problema, una novedosa propuesta pedagógica y una variedad de recursos construidos en el marco del proyecto Cupi2. Las fortalezas del proyecto se presentan basadas en cuatro componentes que se consideran fundamentales para el éxito de este trabajo: aprendizaje incremental, aprendizaje basado en problemas, el uso de herramientas tecnológicas de soporte para la enseñanza y finalmente, la comunidad de enseñanza de la programación que se ha construido para dar soporte a la comunidad académica interesada en la innovación. Al finalizar se muestran algunos resultados obtenidos hasta el momento.

Palabras clave: aprendizaje activo, comunidad de aprendizaje, enseñanza de la programación, objetos de aprendizaje en programación.

ABSTRACT

The difficulties teaching and learning computer programming have been a worldwide recurrent topic for the last twenty years. The community of computer science educators has proposed a vast number of approaches but, unfortunately, none of them has been truly

7 Proyecto ganador Primer Puesto Premio Colombiano de Informática Educativa - Categoría Investigación, Ribie-Col., Universidad del Sinú, Montería. Septiembre 30 de 2009.

8 Profesor Asociado. Director del Departamento de Ingeniería de Sistemas y Computación, Universidad de los Andes. Director del proyecto CUPI2 del grupo de Construcción de *Software* (*TICSw*). Ph.D en Informática, Universidad Joseph Fourier (Grenoble, Francia). Su trabajo de investigación se concentra en los temas de arquitectura empresarial, BPM y *workflows*, arquitectura de *software*, análisis de requerimientos y diseño de *software*. Es promotor de la reforma en la enseñanza y el aprendizaje de programación en educación superior debido a las necesidades actuales en el desarrollo de la industria *software* en Colombia y el desarrollo de nuevos perfiles profesionales en ingeniería basados en la adquisición de habilidades para resolver problemas. E-mail: jvillalo@uniandes.edu.co.

9 Profesora- Instructora. Investigadora del grupo de Construcción de *Software* (*TICSw*) del Departamento de Ingeniería de Sistemas y Computación de la Universidad de los Andes. M.Sc. en Ingeniería de Sistemas y Computación de la Universidad de los Andes. Participa en la divulgación científica del proyecto CUPI2 a través del desarrollo y acompañamiento de la Comunidad Virtual de Profesores de Programación de este mismo proyecto. n-calder@uniandes.edu.co

effective. To the motivational issues suffered by the students, we have to add the absence of a serious study regarding the skills and abilities they have to develop, causing the organization of the programming courses reduced to a follow up of the syntax of a particular programming language. This article presents a new approach to tackle the problem; it is based on a solid conceptual framework and a novel pedagogical methodology, supported by a series of materials and tools that were built in the context of the Cupi2 project. We explain the main aspects of the proposed model focusing on four elements that we consider as fundamentals to the success of this work: incremental learning, problem-based learning, the use of computational tools to support the model and the teaching community we have built. At the end of the article, we summarize the major results obtained so far.

Key words: active learning, learning community, programming, learning objects in programming education.

Recibido: 1 de octubre de 2009
Aceptado: 23 de octubre de 2003

INTRODUCCIÓN

El ciclo básico de formación en programación se encuentra constituido por tres cursos denominados CS1, CS2 y CS3.[10] Este ciclo completo lo toman todos los estudiantes de los programas de pregrado en Ingeniería de Sistemas, otros programas de ingeniería y algunos de programas como Matemáticas, Física o Biología. El curso CS1 suele verse en primer semestre, sin ningún requisito, lo cual implica un potencial desnivel (i. e. diferentes expectativas e intereses personales, diferentes niveles de conocimiento del tema) de los estudiantes que lo toman.

El objetivo de los cursos de programación no es únicamente que el estudiante aprenda a escribir un programa de computador. Estos cursos deben generar una gran cantidad de habilidades en los estudiantes: ellos deben aprender a entender un problema (abstraer, modelar, analizar), a plantear soluciones efectivas (reflexionar sobre una abstracción, definir estrategias, seguir un proceso, aplicar una metodología, descomponer en subproblemas), a manejar lenguajes para expresar una solución (codificar, entender y respetar una sintaxis), a utilizar herramientas que entiendan esos lenguajes (programar, compilar, ejecutar, depurar), a probar que la solución sea válida (entender el concepto de corrección y de prueba), a justificar las decisiones tomadas (medir, argumentar), etc. Estas son habilidades básicas con las que debe contar cualquier profesional en ingeniería.

Desde hace años existe una preocupación en la comunidad académica por estos cursos, debido a la alta tasa de mortalidad, al bajo nivel de motivación de los estudiantes y al alto porcentaje de deserción que lo anterior genera. Se han propuesto algunas herramientas y enfoques pero ninguno parece dar una respuesta integral a la problemática. Las soluciones

10 De acuerdo con: Association for Computing Machinery (ACM), quienes son reconocidos como la asociación mundial más grande en computación científica y educativa.

simples (cambiar de libro, utilizar otro lenguaje de programación, cambiar el orden de los temas, etc.) ya han sido intentadas en infinidad de variantes sin que se logren mejoras efectivas. A lo anterior se suma la creciente preocupación nacional e internacional por la disminución en la demanda de admisión al programa de ingeniería de sistemas.

El proyecto Cupi2 se inició en el año 2004, con el propósito de buscar nuevas maneras de aprender y de enseñar a programar, haciendo énfasis en la motivación de los estudiantes y teniendo en cuenta todos los aspectos involucrados en el problema. Cupi2 se planteó desde un comienzo como un proyecto multidisciplinario de investigación y ha pasado por siete etapas principales: (1) diagnóstico, (2) definición del marco conceptual, las hipótesis, las variables, los indicadores y los instrumentos, (3) diseño de la propuesta, (4) validación de las hipótesis por medio de pilotos controlados, (5) construcción de los materiales y herramientas de apoyo, (6) proceso de formación y acompañamiento de profesores y (7) creación de una comunidad de aprendizaje y difusión de resultados.

Para lograr una solución integral al problema planteado, se considera que se deben tener en cuenta los siguientes aspectos, algunos de los cuales serán tratados con más detalle en este artículo:

- Un modelo pedagógico adaptado al perfil de los estudiantes actuales. Es necesario volver a pensar a fondo la manera de enseñar a programar, utilizando para esto la estructura definida por los ejes conceptuales antes identificados.

- Un modelo de evaluación orientado a la verificación de las habilidades que se han debido generar en los alumnos, que tenga en cuenta los diferentes aspectos que hacen parte de la tarea de programar.

- Un conjunto de materiales y herramientas de soporte al proceso de aprendizaje.

- Un mecanismo de seguimiento a los estudiantes que mida algunas variables que indiquen el éxito (o fracaso) del proceso de aprendizaje. El mecanismo incluye la definición de dichas variables, el diseño de los instrumentos que permitan recoger la información y la definición de los mecanismos que utilicen lo anterior para garantizar un proceso de mejoramiento continuo.

- Un esquema de formación de profesores que les dé una comprensión global de la problemática y que les permita participar de manera efectiva en el proceso de aprendizaje de los estudiantes.

- Un conjunto de herramientas de acompañamiento para los profesores, que faciliten la planeación de las clases, la comunicación con los estudiantes, el registro de los resultados obtenidos, el reporte de las actividades y logros, el seguimiento, etc.

- Una estrategia de difusión de conocimiento para los programas de Ingeniería de Sistemas en Colombia. De acuerdo con dicha estrategia, el proyecto Cupi2 busca aportar al fortalecimiento de la industria de *software* colombiana, a través de la generación y apropiación de conocimiento en comunidad.

Como proyecto de investigación, Cupi2 ha seguido un proceso serio y disciplinado, con resultados medibles y repetibles, y con publicaciones internacionales que reportan las principales hipótesis de trabajo y los mecanismos de validación. Adicionalmente, todos los materiales desarrollados como soporte han sido evaluados y mejorados de manera incremental, hasta lograr en la actualidad un estado estable. El proyecto es actualmente eje de referencia para el cambio curricular de muchos programas de Ingeniería de Sistemas en Colombia.

Dentro de los objetivos de Cupi2 se contempla la propagación de conocimiento y experiencias derivadas del trabajo realizado en la Universidad de los Andes de manera abierta para la comunidad académica. Así, los recursos, estrategias y herramientas desarrolladas son puestos a disposición de estudiantes y universidades interesadas. También se generan trabajos de soporte a la migración y adaptación de currículos en diversas universidades, teniendo como referencia la experiencia de los 4 años de funcionamiento de Cupi2 en los Andes.

MOTIVADORES DEL PROYECTO CUPI2

Evidencias en altas tasas de mortalidad y de estudiantes trabajando bajo condiciones de poca motivación en los cursos de programación [8], revelan la necesidad de enfrentar el problema de la enseñanza y el aprendizaje de la programación desde aproximaciones alternativas. Aunque muchas investigaciones se han presentado con diversas herramientas y estrategias de soporte para enfrentar este problema [1] [13], ninguna de ellas ha presentado resultados suficientes en cuanto a la articulación de objetivos pedagógicos de los cursos y experiencias positivas por parte de estudiantes y profesores.

Lo que a primera vista puede parecer un problema sencillo, como es la enseñanza de la programación de computadores, en realidad es un tema con múltiples facetas que sigue siendo una línea abierta de investigación en gran parte del mundo. Muchos enfoques y herramientas distintas han sido propuestas en los últimos diez años [21] [3] [9], y sin embargo, no parece existir una solución completamente satisfactoria. La aparición del enfoque de objetos, el surgimiento de nuevos lenguajes de programación, herramientas y tecnologías, han introducido nuevos factores de dificultad a la ya compleja labor de enseñar a programar.

En la Universidad de los Andes, el curso de Introducción a la Programación es obligatorio para 10 programas de pregrado (los 8 programas de Ingeniería, Física, Matemáticas), lo que representa una población semestral de cerca de 1.200 estudiantes, repartidos en aproximadamente 30 secciones. Durante un periodo de unos 10 años, se utilizó el lenguaje de programación C, y se incluyeron diversos elementos metodológicos como definición de precondiciones y poscondiciones para la especificación de los problemas. A partir de 1998, se creó una variante de dicho curso, en la cual se introdujeron conceptos básicos de

la programación orientada por objetos y se utilizó el lenguaje de programación C++. La justificación de dicha variación era que los estudiantes (de Ingeniería de Sistemas) debían prepararse de manera diferente para enfrentar los cursos que verían más adelante en el área de Ingeniería de *software*, y que el hecho de introducir desde el comienzo un lenguaje orientado por objetos debería significar una ganancia en el proceso. Aprovechando esta separación, se introdujeron en dicha variante del curso algunos conceptos de control de calidad [7] algunos conceptos básicos de patrones de diseño [5] y algunos conceptos de interfaces gráficas. Los resultados, aunque satisfactorios en algunos aspectos, no llenaron completamente las expectativas que se tenían. Una alta tasa de deserción, unida a una alta tasa de mortalidad y a algunos problemas evidentes de motivación de los estudiantes, demostró que estas estrategias seguían siendo incompletas.

Dificultades en la enseñanza de la programación

Por su misma naturaleza, un curso de enseñanza de la programación está orientado a la generación de habilidades, más que a la simple exposición de un conjunto de conceptos. Si se piensa en la cantidad de teoría que hay detrás de la programación se sorprende del poco número de conceptos distintos y de su simplicidad. Este hecho en sí mismo no es un problema, a menos que el énfasis del curso se haga en la exposición de las definiciones y todo lo demás se vuelva accesorio.

En un curso típico de programación, el profesor, después de presentar en clase los conceptos (o pedir que los estudiantes lean de un libro), responde las preguntas que puedan surgir de dichas definiciones, y luego desarrolla ejercicios en el tablero. Dicha manera de enseñar a programar se basa en la esperanza de que el estudiante sea capaz de detectar patrones en los problemas planteados y los logre asociar con las técnicas que usa el profesor cuando desarrolla un ejemplo en el tablero y que luego, el estudiante sea capaz de hacer la generalización adecuada en su cabeza, para poder aplicar esa asociación patrón-técnica en la resolución de otros problemas. La única técnica que se enseña en los cursos de programación es "dividir y conquistar" y se hace a un nivel de informalidad y vaguedad que no pasa de ser una buena intención.

Este enfoque, que se puede llamar aprendizaje por imitación, se basa en la hipótesis de que el estudiante es capaz de generar las habilidades necesarias tratando de imitar lo que el profesor hace. Esto tiene muchos problemas implícitos y explica una buena parte de las dificultades de enseñar a programar. Como se presenta en [15]:

- No se genera el vocabulario necesario para hablar de la manera, construir un programa. Ver la solución de un problema en un libro, o ver al profesor construyéndola en el tablero, no garantiza la adecuada generación de habilidades en el estudiante. Y aunque en un porcentaje significativo de los casos este enfoque funciona adecuadamente (así se ha enseñado programación durante los últimos 20 años), el problema es que su éxito depende de factores que no controlamos. Esto hace que a los "buenos estudiantes" les vaya bien, y a los "malos estudiantes" les vaya mal, sin que los profesores puedan hacer nada al respecto.

- Se da la sensación de que el "cómo hacer las cosas" no es enseñable, y que es algo que depende de la inspiración y del ingenio del programador.

- La calidad del curso es muy dependiente de la capacidad del profesor para inducir a los estudiantes a aprender por imitación.

- No se controla la generalización que los estudiantes hacen a partir de lo que ven en clase. En muchos casos llegan a establecer patrones incorrectos, que sólo se detectan en cursos más avanzados.

Un curso típico de programación sigue un orden de abajo hacia arriba en la presentación de los conceptos. Se presentan primero los elementos de base (tipos simples, operadores, expresiones, etc.) y luego, poco a poco, se van introduciendo las estructuras de control, la construcción de funciones, el paso de parámetros, los vectores, las matrices, etc. Sólo al final del curso el estudiante es capaz de construir un programa completo. Pocas veces se alcanza a estudiar el tema de interfaces o de pruebas. Como consecuencia:

1. El estudiante no tiene una visión clara de la razón por la cual se introduce un concepto, porque no le ve una necesidad real. Esto dificulta su proceso de aprendizaje.
2. Muchos de los temas que se ven resultan artificiales debido al tamaño de los problemas sobre los cuales se puede trabajar, y se envían mensajes incorrectos con respecto a su verdadera utilidad.
3. Se necesita cubrir una gran cantidad de temas (un par de meses de estudio) antes de poder hacer algo interesante con contexto, y que pueda dar al estudiante la sensación de que lo que está aprendiendo puede ser útil para su vida profesional.

Adicionalmente, la labor de programar implica la introducción de conceptos que pertenecen a distintos ejes temáticos. Se puede decir que para que un estudiante sepa programar debe manejar algunos conceptos y habilidades de los siguientes dominios de conocimiento: modelaje de una realidad, especificación de un problema, lenguajes de programación, uso de herramientas computacionales, algorítmica, metodología de programación, proceso de construcción de un programa (análisis, diseño, pruebas), etc. El estudiante debería entender la programación como el equilibrio de muchos dominios, con relaciones profundas entre ellos, que permiten construir una solución a un problema expresada como un programa de computador.

Un curso de programación puede estar estructurado siguiendo uno o varios de estos ejes temáticos. Los cursos típicos de programación están muy guiados por el eje asociado con el lenguaje, y los demás sólo aparecen puntualmente en algunas partes del curso. Esto hace que el curso (al igual que la mayoría de libros para esos cursos) esté basado en un recorrido de las estructuras sintácticas del lenguaje. Todos los demás aspectos que hacen parte de la labor de programar, se ven de manera lateral, o sencillamente se ignoran.

Esta manera de estructurar el curso tiene como consecuencia que al estudiante no se le dé una visión real de lo que es programar. El estudiante le da entonces mucha más importancia a los

elementos del lenguaje de programación que al proceso de construir un programa (o a cualquier otro eje temático). En clase, por ejemplo, hay muchas más preguntas ligadas al uso de una librería en particular, que a la manera de especificar un problema. El estudiante menosprecia algunos de los elementos que el profesor intenta introducir de manera lateral al curso, y eso dificulta el proceso de enseñanza. Al final, el estudiante tiene la impresión de que aprendió un lenguaje, pero no es consciente ni valora realmente las habilidades que generó en los demás ejes. Por eso, en algunos casos, si el estudiante sabe que no va a utilizar en su vida profesional el lenguaje de programación que vio en el curso, siente que está perdiendo el tiempo.

En la búsqueda de propuestas actuales frente al tema de innovar la enseñanza de la programación, muchos autores concuerdan con la necesidad de integrar la generación de habilidades más que la transmisión de conocimientos planos, especialmente en el nivel de educación superior. [10] [4].

La educación en ingeniería es un campo con fuertes tradiciones pero diversos ejemplos de transición de currículos que utilizan estrategias pedagógicas, que involucran a los estudiantes como agentes de generación de su propio conocimiento, han evidenciado la necesidad de cambio y la fuente de posibles soluciones frente a dificultades en la enseñanza de ingeniería [4].

La necesidad de materiales y herramientas de soporte

Como parte importante de las estrategias que se desarrollan frente al problema de enseñar a programar, se encuentran las herramientas de soporte. Principalmente herramientas de visualización y animación de algoritmos emergen en la década de los ochenta para luego evolucionar en tecnologías gráficas usadas para la representación de diferentes conceptos en computación. Los instructores y profesores han encontrado gran soporte en estas herramientas cuando se mide el impacto que tienen en las sesiones de clase y laboratorios [11]. Estas herramientas involucran a los estudiantes en diversos niveles de compromiso con el aprendizaje. El análisis de impacto de este tipo de estrategias sobre el proceso de aprendizaje, ha resultado en experiencias que prueban que aún con el desarrollo de las mejores estrategias, éstas sólo acarrean aprendizaje cuando se utilizan en un ambiente adecuado [11].

La generación de conocimiento en comunidad

Finalmente, frente a la necesidad de enfrentar cambios y manejar conocimiento de experiencias previas, muchas redes sociales virtuales han aparecido en ambientes de enseñanza desde de la academia hasta los sectores privados.

Sin embargo, estas redes están principalmente orientadas sobre elementos de interacción social, más que en el soporte directo que se pueda ofrecer al proceso de aprender cómo enseñar a programar. Un ejemplo de este tipo de red en línea es el sitio *web TeachingToday* [14]. Como los desarrolladores del sitio *web* son conscientes de que las comunidades en línea proporcionan una abundancia de oportunidades a los educadores para compartir estrategias de enseñanza, ideas y mejores prácticas, se elaboró un almacén en línea de consejos,

herramientas y recursos que son fáciles de utilizar y llamativos a nivel pedagógico. Estos elementos también pueden encontrarse en la Red de Educación National Geographic [12]. Esta comunidad incluye información acerca de oportunidades de desarrollo profesional, tales como talleres y eventos en línea a los que los miembros pueden asistir. Una de las más prometedoras redes en línea es desarrollada en el sector privado por *Yahoo: Yahoo Teachers* [20]. Esta red, diseñada por y para los profesores, proporciona un lugar en el que los docentes de todos los campos pueden crear, modificar y compartir los planes de estudio basados en estándares.

METODOLOGÍA DEL PROYECTO CUPI2

El modelo planteado por el proyecto Cupi2 tiene una sólida base conceptual e integra soluciones novedosas a las distintas facetas involucradas en el problema. El proyecto Cupi2 ha logrado articular su propuesta alrededor de seis dimensiones de un sólido marco conceptual. Estas dimensiones incluyen: (1) un modelo pedagógico, (2) un conjunto de materiales de soporte al proceso de aprendizaje de los estudiantes, (3) un conjunto de mecanismos de seguimiento y mejoramiento continuo, (4) un modelo de evaluación, (5) un proceso de formación de profesores y (6) un conjunto de herramientas de apoyo a los profesores que incluyen la creación de una comunidad de aprendizaje y difusión de resultados.

Marco conceptual en el diseño de los cursos

El marco conceptual está definido sobre los ejes alrededor de los cuales gira la tarea de programar. Estos son: (1) modelaje y solución de problemas, (2) algorítmica, (3) tecnología y programación, (4) herramientas, (5) procesos de *software*, (6) arquitectura, y (7) técnicas de programación y metodologías. En cada uno de estos ejes hay conocimientos y habilidades que se deben incluir como parte de los cursos, a distintos niveles de profundidad (Figura 1). Los cursos se encuentran divididos en niveles incrementales en los que se exploran diferentes conocimientos y habilidades, al tiempo que se refuerzan permanentemente los objetivos adquiridos en niveles anteriores. En cada uno de los cursos y en cada uno de los niveles se encuentran elementos de los 7 ejes conceptuales mencionados anteriormente, lo que permite la generación integral de conocimientos y habilidades (Figura 1).

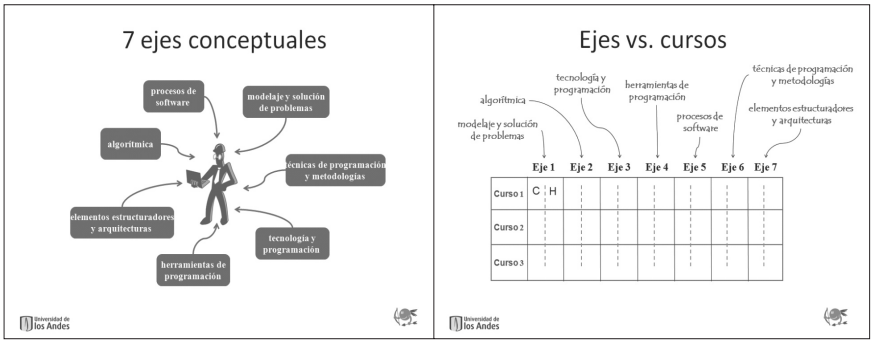

Figura 1. Ejes conceptuales y su distribución en los cursos de programación

El modelo pedagógico del proyecto Cupi2 se encuentra soportado por cuatro componentes que permiten la construcción de una solución balanceada frente problema de aprender a programar. Cada uno de estos componentes ha sido validado en experiencias mundiales de enseñanza de diferentes campos de la ciencia, teniendo importante impacto en la educación en ingeniería como se presenta en [4]. A continuación se describen sus principales características.

Aprendizaje Activo

Este es el primer punto del modelo pedagógico propuesto. Puesto que se trata de un curso donde la generación y desarrollo de habilidades es fundamental, el estudiante debe asumir un papel central en el proceso: debe ser el protagonista principal del curso. Bajo el modelo de aprendizaje activo, los estudiantes se comprometen con su propio aprendizaje. Esto cambia por completo la estructura de las clases y laboratorios, y la manera como el profesor participa en el proceso de aprendizaje del estudiante. El reto del profesor es, entonces, definir un conjunto adecuado de escenarios en los cuales el estudiante se sienta motivado para trabajar, con objetivos claros y ligados consistentemente con la evaluación del curso.

Estos escenarios se construyen de manera que los estudiantes participan en las clases no sólo como receptores pasivos. La lectura, discusión, y proposición de soluciones, involucra al aprendiz en actividades de alto nivel como análisis, síntesis y evaluación [2].

Aprendizaje Basado en Problemas y Proyectos (PBL)

Este es el segundo punto del modelo pedagógico propuesto. Se espera que los estudiantes enfrenten dificultades derivadas de la baja motivación, a través del aprendizaje basado en problemas que reflejan retos del mundo real. De esta manera, el conocimiento y el desarrollo de habilidades se logra por medio de la relación que existe en partes específicas de un problema propuesto.

Al comienzo de cada nivel el profesor presenta el problema o ejercicio sobre el que se va a trabajar. La diferencia entre lo que el estudiante sabe en ese momento y lo que el estudiante necesita para resolverlo, es lo que lo motiva a estudiar los temas nuevos. Al final del nivel, la evaluación se hace en tres partes: la revisión del ejercicio que cada estudiante entrega, un examen escrito sobre el ejercicio y un examen práctico, que consiste en una extensión al programa entregada por el estudiante. Las extensiones se describen como nuevas funcionalidades que se pueden agregar al sistema, la incorporación de nuevos componentes gráficos o cambios en los algoritmos implementados. De esta manera, el ejercicio sirve tanto para determinar el objetivo del nivel, como para evaluar los resultados obtenidos. Debido a que los problemas siempre se escogen de manera que tengan un contexto real, el estudiante se ve obligado a comprender un dominio particular, a leer un texto y entenderlo, y a manipular abstracciones del mismo para resolverlo.

Aprendizaje Incremental

Este es el tercer punto del modelo pedagógico propuesto. Cada uno de los tres cursos del ciclo de programación se encuentra dividido en 6 niveles, en cada uno de los cuales se introducen conceptos nuevos en algunos de los ejes, se refuerzan conceptos vistos en niveles anteriores y se aplican todas las habilidades generadas hasta el momento en la solución de uno o varios problemas.

Esto remplaza la aproximación tradicional en la que se estudian los temas de abajo hacia arriba, hasta lograr tener todos los elementos necesarios para el desarrollo de un programa. En nuestro enfoque, el estudiante, desde el primer día de clase, comienza a programar, utilizando a nivel elemental algunos elementos de los lenguajes de programación y siguiendo un proceso que lleva desde la etapa de análisis hasta la etapa de implementación. Así como los conceptos se van introduciendo de manera gradual y a distintos niveles de profundidad, el estudiante va desarrollando las habilidades de forma incremental, respetando siempre la estructura de la solución.

En los primeros niveles el estudiante aprende a identificar los distintos componentes de un programa, a leerlos, a entenderlos y a completar las partes que faltan para lograr que el programa funcione completamente. Para esto se trabaja sobre programas incompletos, cuyas partes faltantes se pueden construir con los conceptos y habilidades estudiados en el nivel.

Con la aproximación propuesta se logra avanzar mucho más en cada curso y se llegan a construir programas más complejos, que están lejos de lo que se obtiene en un curso tradicional de programación, con resultados a la vez superiores en cuanto a motivación y notas obtenidas por los estudiantes.

Aprendizaje basado en ejemplos

El último componente es el modelo de aprendizaje basado en ejemplos. De acuerdo con este modelo, los estudiantes tienen acceso a ejemplos de buenas prácticas y soluciones comunes a problemas. El proyecto Cupi2 actualmente provee más de 150 ejemplos implementados y probados para sus estudiantes.

Tecnología y herramientas en el modelo Cupi2

La implementación de la aproximación propuesta requirió que se construyera una gran cantidad de materiales de apoyo, dentro de los cuales se encuentran: (1) los libros [18, 19], (2) el depósito de problemas y ejemplos, (3) los entrenadores, (4) los tutoriales y laboratorios y (5) los mapas mentales y videos expositivos.

Los libros fueron publicados por una editorial internacional, para garantizar la difusión del proyecto en toda Latinoamérica. En el depósito de problemas se tiene en este momento más de un centenar, dirigidos a los tres cursos de programación. Los problemas están resueltos y sirven como base para proponer los ejercicios que deben resolver los estudiantes. Los tutoriales y talleres

permiten al estudiante trabajar de manera autónoma sobre algunas herramientas y problemas para resolver un conjunto de retos planteados. Los entrenadores, por su parte, admiten que el estudiante manipule conceptos abstractos a través de metáforas, de herramientas de animación de algoritmos y de juegos (Figura 2). Estos entrenadores se han validado como instrumentos de generación de habilidades y el resultado de esta evaluación se hizo público en [17]. Actualmente se cuenta con cerca de cuarenta entrenadores para los tres cursos de programación. Todos los materiales mencionados son de acceso público desde el portal del proyecto además de publicarse periódicamente en el Banco de Objetos de Aprendizaje Colombia Aprende (http://www.colombiaaprende.edu.co/objetos/), utilizando como estándar de descripción el modelo IEEE LOM.

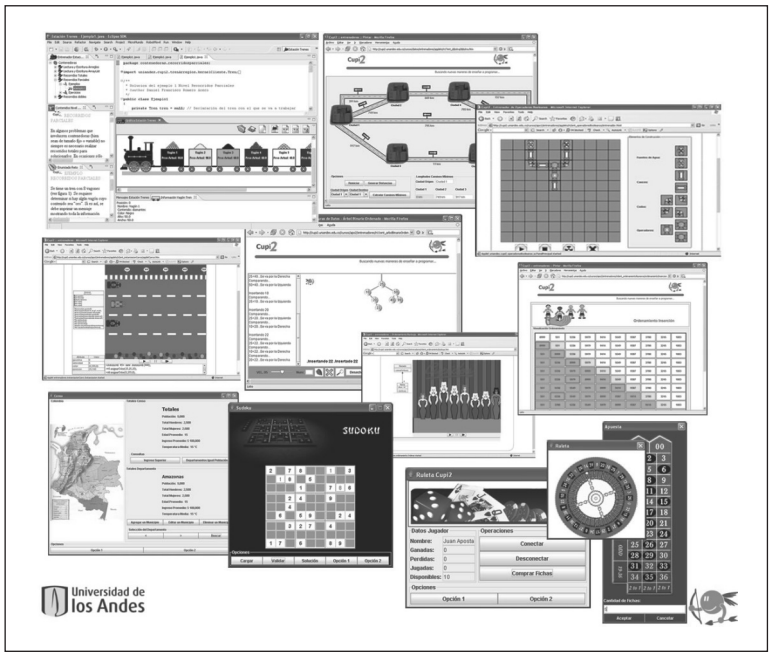

Figura 2. Ejemplos, Entrenadores y Ejercicios

Adicionalmente, como parte de la propuesta innovadora de Cupi2, cada semestre se incorporan nuevos recursos de apoyo para el diseño de clases y la presentación de los niveles de los cursos de programación. Estos recursos buscan responder de forma creativa a problemas específicos de cada nivel y ser una forma novedosa de presentar ejemplos concretos que alimentan las explicaciones de clase.

Muchos de estos recursos hacen parte del proceso de difusión del proyecto, permitiendo a los profesores nuevos o aquellos que desean hacer parte de la comunidad Cupi2, la presentación de algunos de los miembros actuales. Como ejemplo de los materiales incluidos en el último año se contempla videos, mapas mentales y un buscador sintáctico sobre los mapas.

El desarrollo de una solución integral

El enfoque propuesto en Cupi2, más que una solución terminada, representa un espacio de soluciones posibles, adaptables a los contextos y necesidades de los profesores de las distintas universidades. Cada profesor, teniendo en cuenta los objetivos pedagógicos de cada nivel en cada eje, y contando con todos los recursos del proyecto, debe diseñar una estrategia de trabajo con sus estudiantes. Se evita toda camisa de fuerza para los profesores y se los hace partícipes de un proceso de aprendizaje colectivo, soportado por un depósito de conocimientos, por un conjunto de herramientas de apoyo a la preparación de clase, por una serie de tutoriales de inducción y acompañamiento y por mecanismos de comunicación entre ellos.

Como se discutió anteriormente, existen diversas dificultades en la enseñanza de la programación que han promovido el uso de novedosas estrategias pedagógicas. Sin embargo, los cambios en planes de educación requieren más que el trabajo alrededor del proceso de aprendizaje de los estudiantes. Actores como profesores y facultades de ingeniería, soportan la administración del conocimiento que se genera en las comunidades académicas y, en consecuencia, el mejoramiento en la enseñanza. Como solución frente al reto de soportar estos diversos actores que participan en la enseñanza de la programación, el proyecto Cupi2 propone la integración de una comunidad de enseñanza que permita el intercambio de estrategias, centralice el conocimiento grupal y ofrezca soporte a las necesidades de comunicación que esto conlleva.

La comunidad virtual de profesores Cupi2 es la red social de educadores (a nivel nacional e internacional) que se apoyan en el proyecto Cupi2 para el diseño y desarrollo de sus cursos de programación. La comunidad se encuentra soportada por un conjunto de herramientas que permiten la interacción de los miembros, el intercambio de materiales de trabajo y la unión de experiencias de cada participante.

Esta comunidad nace como respuesta a la necesidad de la difusión de conocimiento y estrategias que se desarrollan alrededor del proyecto Cupi2. Dentro del proyecto se contemplan las conferencias, capacitaciones, acompañamiento y planes de tutoría para universidades interesadas en la reforma de la enseñanza de la programación [6]. Como consecuencia, decenas de charlas se han realizado en diferentes regiones colombianas, actualmente se encuentran activos varios programas de acompañamiento a universidades que quieren incluir el proyecto en su currículo y, finalmente se encuentra la comunidad virtual para que las fronteras geográficas no sean límite en la difusión de experiencias que los diferentes participantes e interesados quieren aportar.

Actualmente, la comunidad Cupi2 integra más de 25 universidades en Colombia y algunos institutos en el resto del mundo, en donde la enseñanza de la programación se realiza a través de la metodología Cupi2. El intercambio de las experiencias de cada uno de nuestros miembros, permite la continuidad, evolución y crecimiento del proyecto.

A continuación se resumen las principales características de la comunidad virtual de profesores y las herramientas de soporte para los procesos de inducción, acompañamiento, diseño de niveles y publicación de experiencias que permiten la dinámica de difusión y apoyo a los profesores participantes en este proyecto. Una visión más amplia de esta comunidad se describe en [16].

El Portal Cupi2

El sitio *web* de Cupi2 (http://Cupi2.uniandes.edu.co) es una herramienta que integra el trabajo de difusión del proyecto, la herramienta de publicación semestral de trabajo para los estudiantes, las herramientas de soporte administrativo y la entrada al portal de la comunidad virtual.

En el portal converge el trabajo de todos los actores en Cupi2. Existen ejemplos, ejercicios, entrenadores, tutoriales y videos de apoyo, entre otros recursos, disponibles para los estudiantes. Para los profesores hay espacios exclusivos que les ayuda en la tarea de planear y organizar cursos. También provee espacios para coordinadores que se encargan de administrar y asegurar la disponibilidad de los recursos a la comunidad.

Depósito de recursos

Uno de los propósitos de la comunidad es permitir el intercambio de materiales y experiencias que contribuyan nuevas ideas, ejemplos, ejercicios y todo tipo de material para el diseño de clases y la inducción de nuevos profesores.

Actualmente el depósito de la comunidad cuenta con:

1. Depósito de bitácoras, en donde los profesores registran en cada nivel del curso la preparación de las clases y los resultados obtenidos. Esta información está abierta para consultas de otros profesores. Se encuentran registradas en este momento cerca de 1.000 bitácoras (desde 2005-1), clasificadas por profesor, curso y nivel.
2. Depósito de materiales, en donde los profesores dejan copia de los enunciados de sus exámenes, hojas de trabajo desarrolladas en clase, presentaciones, enunciados de laboratorios, etc. Esta información está abierta para consultas de otros profesores. En este momento se encuentran registrados más de 4.200 documentos.
3. Depósito de materiales de inducción para la formación de profesores nuevos.

Todos estos recursos se encuentran clasificados y presentados por cursos, niveles, tipo de material, zonas de trabajo (profesores o estudiantes) en el portal de Cupi2.

Secuencias de aprendizaje

Las secuencias de aprendizaje son el diseño del conjunto de actividades y recursos con los que un profesor organizará cada nivel de su curso. Todos los miembros de la comunidad de profesores pueden consultar todos los depósitos de recursos y proponer las actividades de cada

clase. Al finalizar las clases pueden registrar la bitácora con el resultado de su experiencia. Cada secuencia sirve de apoyo a los profesores que se enfrentan al reto de dictar estos cursos por primera vez y además como registro histórico de experiencias (Figura 3).

Espacios y cursos

Como herramienta para la organización de documentos, los profesores de la comunidad cuentan con espacios virtuales para almacenamiento de sus recursos. Estos espacios se diseñan para mantener la estructura de los recursos de la comunidad de tal manera que siempre se pueda encontrar el material necesitado.

Buscador de recursos

El buscador de recursos y secuencias de aprendizaje permite la consulta de todos los materiales, bitácoras, diseños de clase, etc., que se han registrado en la comunidad a través de los espacios, de los cursos y de las secuencias de aprendizaje. De esta manera los nuevos integrantes tienen acceso fácil y rápido al historial de experiencias y materiales útiles para el diseño de sus clases (Figura 3).

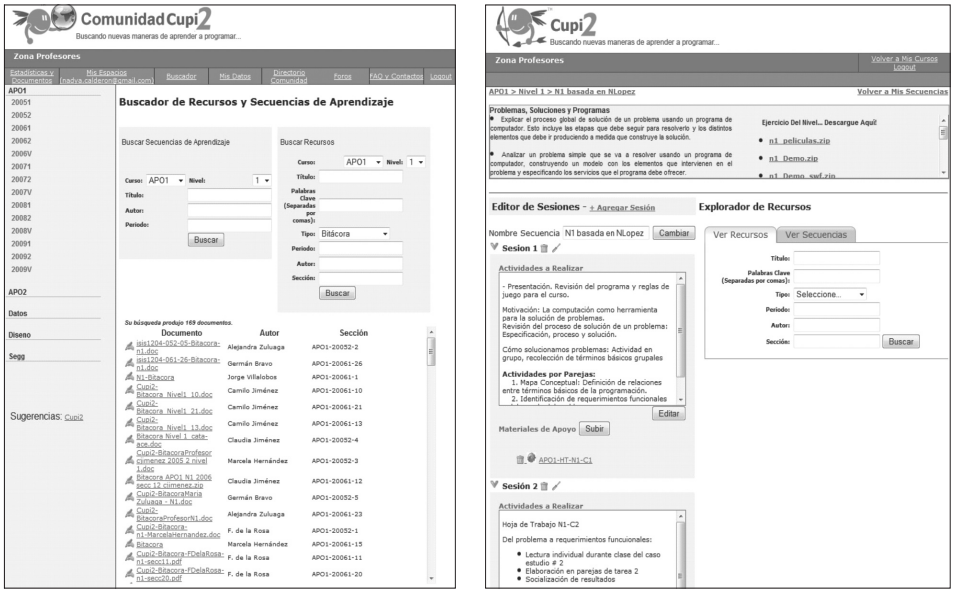

Figura 3. Herramientas para la Comunidad de Profesores

Estadísticas

La comunidad tiene acceso al sistema de información con resultados estadísticos de los cursos del ciclo básico de programación desde 2003. Hay una base de datos con toda la información necesaria para hacer el seguimiento de los indicadores. Es posible, por ejemplo, hacer todo el seguimiento teniendo en cuenta el programa de los estudiantes que toman el curso.

RESULTADOS

Para el seguimiento del proyecto se definieron 4 indicadores principales: (1) los resultados de la encuesta que la Universidad de los Andes hace a los estudiantes cada semestre, en la cual se evalúan 11 preguntas con respecto al profesor. (2) Los resultados de la misma encuesta en la cual se incluyen 6 preguntas sobre el curso, (3) la nota promedio obtenida por los estudiantes del curso y (4) el porcentaje de estudiantes que pierde el curso.

Comparando los valores históricos con el promedio de los semestres en los últimos dos años, donde los resultados son comparables porque tienen el mismo tipo y volumen de estudiantes, se pudo concluir que:

- La evaluación que los estudiantes hacen sobre la percepción de los profesores aumentó casi en un 6%.
- La evaluación que los estudiantes hacen sobre la percepción del curso, aumentó en más de un 20%.
- La nota promedio de los estudiantes aumentó en más del 11%.
- El número de estudiantes que pierde el curso disminuyó en un 50%.

Adicionalmente, para resaltar el crecimiento del proyecto y el impacto que este ha tenido desde su inicio, a continuación se describen algunas métricas de tamaño (calculadas hasta diciembre de 2008) tomadas de [16]:

La Tabla 1 muestra el impacto del proyecto en número de estudiantes, profesores y universidades que se han involucrado desde el 2005. A continuación se mide el tamaño del proyecto en término de los recursos de aprendizaje que se encuentran almacenados en el repositorio de la comunidad de profesores, los cuales son el resultado de las contribuciones realizadas por miembros de esta comunidad y el desarrollo de ejercicios y ejemplos en la Universidad de los Andes (Tabla 2).

Tabla 1. Impacto del Proyecto Cupi2 - Número de Participantes

Número de universidades inscritas en la comunidad	**30**
Miembros activos en la comunidad	**157**
Total número de profesores involucrados desde 2005	**198**
Total número de estudiantes involucrados desde 2005	**6.798**

Tabla 2. Tamaño del Proyecto: Número de Recursos de Aprendizaje

Ejemplos	**78**
Ejercicios para laboratorio	**106**
Hojas de trabajo	**821**
Tutoriales en tecnología	**15**
Videos , animaciones y demos	**172**
Entrenadores	**33**
Mapas Mentales	**108**
Exámenes (Escritos y Laboratorios)	**2.959**
Implementaciones de Estructuras de Datos	**52**

Como tercera métrica se presenta el tamaño del proyecto desde la perspectiva de desarrollo de *software*. Alrededor de medio millón de líneas de código y más de 1.000 documentos se han construido para los ejemplos, entrenadores, ejercicios, tutoriales y herramientas de soporte (Tabla 3). Como medida del crecimiento de la comunidad de profesores, se muestra su tamaño en términos del número de bitácoras de experiencias y secuencias de aprendizaje que han sido desarrolladas por los profesores que participan activamente en ella, y que sirven como soporte para el ingreso de nuevos miembros continuamente (Tabla 4).

Tabla 3. Tamaño del Proyecto: Desarrollo de *Software*

LOCs : ejemplos, ejercicios de laboratorio, entrenadores	**321.036**
LOCs de pruebas : ejemplos, ejercicios de laboratorio, entrenadores	**82.963**
Documentos de especificación de ejemplos y ejercicios, especificaciones de requerimientos funcionales, modelos UML.	**1.138**
Herramientas de soporte para desarrolladores	**10**

Tabla 4. Tamaño de la Comunidad: bitácoras y secuencias de aprendizaje

Número de secuencias de aprendizaje disponibles en la comunidad	**434**
Documentos con bitácoras de experiencias de clase	**974**

Finalmente se describe el impacto del proyecto calculando el número de visitas del portal Cupi2. Debido a que los recursos del proyecto se encuentran actualmente en español, y a que la mayoría de los miembros de la comunidad están distribuidos en universidades colombianas, los datos obtenidos describen este mismo fenómeno (Tabla 5).

Tabla 5. Medida de Impacto: Visitas al portal Cupi2 en el 2008

PAÍS	PROMEDIO DE VISITAS MENSUAL 2008	TOTAL VISITAS REPORTADAS 2008
Colombia	16.577	156.884
España	1.302	13.154
Estados Unidos	685	6.695
Otros	167	4.364

Adicionalmente, como resultado del objetivo inicial de Cupi2 para convertirse en fuente de conocimiento y estrategia de difusión en la educación de la programación en Colombia, se han realizado decenas de conferencias, programas de acompañamiento, intercambio de profesores y estudiantes quienes participan como actores directos del proyecto en las instalaciones de la Universidad de los Andes. El trabajo desarrollado alrededor de las estrategias de apropiación de conocimiento se encuentra descrito en [6].

CONCLUSIONES

Una de las principales conclusiones a las que se llegó en la etapa de diagnóstico es que el problema de enseñar a programar es complejo y que cualquier solución que se proponga debe contemplar soluciones integrales a los distintos componentes del problema. El proyecto Cupi2 ha logrado articular su propuesta alrededor de seis dimensiones principales. Estas dimensiones incluyen: (1) un modelo pedagógico, (2) un conjunto de materiales de soporte al proceso de aprendizaje de los estudiantes, (3) un conjunto de mecanismos de seguimiento y mejoramiento continuo, (4) un modelo de evaluación, (5) un proceso de formación de profesores y (6) un conjunto de herramientas de apoyo a los docentes. Es en el equilibrio y sincronización de todo lo anterior, lo que el proyecto Cupi2 ha logrado construir como una solución exitosa para enfrentar el problema de enseñar a programar.

Con respecto al impacto del proyecto en el desarrollo de la carrera de Ingeniería de Sistemas, es claro que los cursos de programación son fundamentales en la formación de ingenieros. Ante la rápida evolución que debe soportar la carrera, es indispensable contar con una base muy sólida en cuanto a las habilidades necesarias para resolver problemas usando un computador. Con el proyecto Cupi2 se ha logrado que los cursos de programación soporten mejor al resto de cursos del currículo, dando al estudiante una visión global de la problemática de construcción de *software* y permitiendo obtener resultados que antes se encontraban restringidos a cursos más avanzados. La mortalidad en los cursos ha disminuido hasta en un 50%, a la vez que la motivación de los estudiantes (medida a través de las encuestas de la Universidad) ha aumentado en más del 20%. Los estudiantes terminan los cursos con una actitud mucho más positiva con respecto a lo que quiere decir desarrollar *software* y con una visión global de todas las oportunidades que allí se pueden encontrar.

Adicionalmente, todos los resultados del proyecto Cupi2 son públicos (http://Cupi2.uniandes.edu.co) y se encuentran apoyando en este momento a un buen número de universidades colombianas. Otra importante conclusión del trabajo realizado en los últimos 5 años de proyecto es el impacto en la comunidad académica que Cupi2 puede soportar.

En este momento los libros desarrollados [18,19] en el marco del proyecto se encuentran a la venta en 11 países de Latinoamérica. Se espera, por esta razón, que el impacto del proyecto trascienda las fronteras del país y se consolide a nivel latinoamericano. Incluso algunos materiales se encuentran en proceso de traducción al idioma inglés con el fin de lograr una mayor difusión de los mismos.

Dada la importancia que tiene la industria de *software* para un país como Colombia, el proyecto Cupi2 tiene también el potencial de convertirse en un apoyo estratégico para las empresas. Entre más sólida sea la formación de los ingenieros de sistemas, más competitivos serán a nivel internacional, y crearán mayores oportunidades tanto para ellos como para las empresas en las que trabajen.

Frente al carácter creativo e innovador de la solución que se presentó, con el proyecto Cupi2 se hace un aporte novedoso y significativo a un problema compartido por la mayoría de universidades. El diseño vertical de los cursos permite que se introduzcan desde muy temprano algunos temas y habilidades fundamentales, que en los currículos actuales se introducen muy tarde en la carrera. Este cambio tiene un impacto directo en la profundidad de la formación que se logra. Para conseguirlo, Cupi2 hace un uso efectivo de las nuevas tecnologías y de las nuevas tendencias en enseñanza para proponer una solución innovadora, abierta a toda la comunidad.

Finalmente, Cupi2 acaba de completar 4 años de trabajo continuo. En él han participado profesionales de distintos dominios. Se han invertido cerca de mil millones de pesos en la construcción de la solución. Se han desarrollado más de 500.000 líneas de código en la implementación de las herramientas de apoyo y en la comunidad de aprendizaje. Se cuenta en el momento con más de 400 objetos de aprendizaje entre tutoriales, laboratorios, *frameworks*, entrenadores y ejemplos. Más de 1.200 estudiantes por semestre y 45 profesores utilizan los resultados del proyecto en la Universidad de los Andes. Se han dictado conferencias de difusión en universidades de todas las regiones de Colombia.

Todo el *software* desarrollado es en sí mismo un ejemplo de calidad: se siguen estándares claros, se cuenta con pruebas automáticas unitarias, se tienen documentos detallados de análisis y diseño, se hace un uso correcto de los lenguajes de programación y de modelado, se sigue un proceso de inspección y validación, etc. Cada herramienta de entrenamiento cuenta con un manual de uso e instalación (en los casos en los que esto aplica), es fácil de utilizar y es efectivo en el momento de generar las habilidades necesarias en los estudiantes.

Cupi2 es un proyecto con resultados exitosos, con un enorme potencial en Latinoamérica y con un claro impacto positivo en cualquier currículo de ingeniería de sistemas. Puesto que todos los materiales desarrollados son de acceso público, está previsto que beneficie a

toda la comunidad académica latinoamericana como ha venido haciéndolo en la comunidad colombiana.

REFERENCIAS BIBLIOGRÁFICAS

[1] BLUEJ. The Interactive Java Environment. Disponible en: http://www.bluej.org/. (Consultado en Mayo de 2008).

[2] BONWELL C, EISON J. (1991). Active Learning: Creating Excitement in the Classroom. Jossey-Bass publishers.

[3] CASSOLA, E. Elaboración de material educativo para la formación de profesionales en desarrollo de *software*. Congreso Iberoamericano de Educación Superior en Computación (CIESC), Conferencia Latinoamericana de Informática (CLEI). Perú, 2004.

[4] DE GRAFF, E., KOLMOS, A. Management of Change. Implementation of Problem-Based and Projec-Based learning in Engineering. Sense Publishers 2007.

[5] GAMMA, E., HELM R., JOHNSON R., AND VLISSIDES J. Design Patterns: Elements of Reusable Object-Oriented *Software*. Addison-Wesley. 1995.

[6] HERRERA, J., GIRALDO, O. Apropiación de Conocimiento en Instituciones de Educación Superior Privadas: Factores Inhibidores y Potencializadores, Paradigma: Revista Electrónica en Construccción de *Software*, Bogotá - Colombia, Noviembre 2008.

[7] HUMPHREY, W.S. Introduction to the Personal *Software* Process. SEI Series In *Software* Engineering. Addison-Wesley. 1997.

[8] JENKINS T. (2001). The motivation of students of programming. Proceedings of the 6th annual conference on Innovation and technology in computer science education, Canterbury, United Kingdom 2001.

[9] KÖLLING, M. The Problem of Teaching Object-Oriented Programming, Part 1: Languages. Journal of Object-Oriented Programming. Vol. 11, No. 8, pp 8-15. 1999.

[10] LOPEZ, M., WHALLEY, J., ROBBINS, P., AND LISTER, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. Proceedings of the Fourth international Workshop on Computing Education Research (Sydney, Australia, Septiembre 06 - 07, 2008)

[11] NAPS, T. L., ROBLING, G., ALSTRUM, V., DANN, W., FLEISCHER, R., HUNDHAUSEN, C. , KORHONEN, A., MALMI, L., MCNALLY, M., RODGER, S., AND VELÁSQUEZ-ITURBIDE. Exploring the role of Visualization and Engagement in Computer Science

Education. En: Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education, páginas 131-152. New York, NY, USA, 2002.

[12] NATGEO. National Geographic Education Network – Workshops, Forums and more. Disponible en: http://www.ngsednet.org/. (Consultado en Noviembre de 2008).

[13] ROBOCODE. The open source educational game. Disponible en http://robocode.sourceforge.net/. (Consultado en Mayo de 2008).

[14] TEACHINGTODAY. Teaching Tips, lesson plans and more, disponible en: http://teachingtoday.glencoe.com/. (Consultado en Noviembre de 2008).

[15] VILLALOBOS, J.A., CASALLAS, R., MARCOS, K. El Reto de Diseñar un Primer Curso de Programación de Computadores. XIII Congreso Iberoamericano de Educación Superior en Computación, Cali, Colombia, Octubre 2005.

[16] VILLALOBOS, J.A., CALDERÓN N.A., JIMÉNEZ, C.H. CUPI2 COMMUNITY – Promoting a Networking Culture that Supports the Teaching of Computer Programming. International Conference on Computer Supported Education (CSEDU), Portugal, Marzo 2009.

[17] VILLALOBOS, J.A., CALDERÓN N.A., JIMÉNEZ, C.H. Developing Programming Skills by Using Interactive Learning Objects. 14th Conference on Innovation and Technology in Computer Science Education (ITICSE), Francia, Julio 2009.

[18] VILLALOBOS, J.A., CASALLAS, R., Fundamentos de Programación. Aprendizaje Activo Basado en Casos. Editorial Prentice Hall. 2006.

[19] VILLALOBOS, J.A., Introducción a las Estructuras de Datos. Aprendizaje Activo Basado en Casos. Editorial Prentice Hall. 2008.

[20] YAHOO TEACHERS. Yahoo Teachers! Your passion, our technology, disponible en: http://teachers.yahoo.org/. (Consultado en Noviembre de 2008).

[21] ZHU, H. AND ZHOU, M. Methodology First and Language Second: A Way to Teach Object-Oriented Programming. OOPSLA'03. Anaheim, CA., october 2003.