Revista Unimar
Número 57
Julio 2011, 23-32
ISSN 0120-4327
ISSN -E 2216-0116

USO DE HERRAMIENTAS INFORMÁTICAS COMO ESTRATEGIA PARA LA ENSEÑANZA DE LA PROGRAMACIÓN DE COMPUTADORES*

USE OF COMPUTER TOOLS AS A STRATEGY FOR TEACHING COMPUTER PROGRAMMING

Alejandra Zuleta Medina**
Docente Institución Universitaria Centro de Estudios Superiores María Goretti, I. U. Cesmag, Pasto, Colombia

Anívar Chaves Torres***
Docente Universidad Nacional Abierta y a Distancia, UNAD, Pasto, Colombia

El fin primordial de la educación es formar hombres
que sean capaces de hacer cosas nuevas,
y no de repetir lo que otros han dicho y hecho;
formar creadores, inventores y descubridores.
E. Fromm

Fecha de recepción:
22 de febrero de 2011
Fecha de aprobación:
31de agosto de 2011

Palabras claves:
Aprendizaje, enseñanza, programación, recursos.

RESUMEN

Desde mediados del siglo anterior la Programación de computadores se ha convertido en un campo de interés científico e industrial, componente fundamental en la formación actual de Ingenieros de Sistemas, Ingenieros en Computación e Ingenieros Informáticos, y componente complementario en muchos otros programas de ingeniería.

No obstante su importancia en la enseñanza y aprendizaje, y según diversas investigaciones, aún hay muchas dificultades, entre las cuales sobresalen por su recurrencia: dificultad para el diseño, verificación y análisis de complejidad de los algoritmos, aplicación de la recursividad y manejo de memoria dinámica para la implementación de estructuras de datos. Además de la complejidad propia de las asignaturas de programación, se encuentra la falta de conocimiento sobre técnicas de resolución de problemas, aunada a la inconstancia y el bajo interés hacia el estudio por parte de no pocos estudiantes.

En este artículo se presenta el resultado de una revisión crítica de varias investigaciones realizadas en este campo, particularmente las que se basan en la utilización de herramientas informáticas para intentar solucionar algunos de los múltiples problemas que se presentan en el proceso de enseñanza y aprendizaje de la programación.

* Artículo de Investigación Científica y Tecnológica.
** Ingeniera de Sistemas por la Universidad de Nariño.Correo electrónico: alejazul07@gmail.com
***Ingeniero de Sistemas por la Universidad Mariana. Correo electrónico: anivarchaves@yahoo.com

Key words:
Learning, education, programming, resources.

ABSTRACT

Since mid last century, Computer programming has become a field of scientific and industrial interest, a key component in training and computational engineers, and a complementary component in many other engineering programs.

Despite its importance, according to several studies, there are still many difficulties in its teaching and learning, among others: difficulty in design, verification and complexity analysis of algorithms and implementation of recursion and memory management implementing dynamic data structures. In addition to the inherent complexity of programming subjects, it is the lack of knowledge about problem solving techniques, coupled with the inconstancy and low interest in studying of many of the students. This article presents results of critical reviewing about several investigations developed in this area, especially those based in computing tools utilization for trying to solve some of multiples problems associated with process teaching and learning programming.

Hace 50 años la Programación de computadores era considerada un arte; en la actualidad es un área de estudio que comienza en el bachillerato y continúa en algunos programas universitarios como Ingeniería de Sistemas, Informática, Computación y muchos otros. Hoy en día, es una disciplina que cuenta con abundante literatura, herramientas y técnicas que facilitan la producción de software; no obstante, programar implica el desarrollo de procesos mentales complejos que hacen difícil su aprendizaje, tal como lo había considerado Wirth (1989) quien desarrolló el lenguaje Pascal, pensado para enseñar programación.

La dificultad en el aprendizaje de la programación se evidencia en el bajo promedio que obtienen los estudiantes en asignaturas de programación y en el alto índice de los que las repiten, quienes, según la investigación de Timarán et al (2009b) superan el 26%. Y aunque han sido desarrollados abundantes estudios sobre el tema, al parecer aún no se logra una solución satisfactoria, quizá porque los trabajos se ocupan de ciertos elementos curriculares, pero no de la integralidad del problema.

En lo que sigue de este documento, se presenta los resultados de esta revisión organizada en tres partes: en la primera se identifica las principales dificultades que existen en el proceso de enseñanza y aprendizaje de la programación; en la segunda, se hace una reflexión sobre las propuestas que han sido diseñadas para atender dichas dificultades y los resultados alcanzados; y, en la tercera se indica las conclusiones generadas con base en esta revisión.

La Problemática de la Enseñanza de la Programación

La enseñanza de la Programación no se limita a la explicación de conceptos y teorías, sino que busca desarrollar habilidades para trabajar con conceptos abstractos, como por ejemplo la manipulación de datos mediante un algoritmo, la sintaxis y la semántica de los lenguajes de programación y el metalenguaje utilizado para escribir algoritmos y programas. Las investigaciones de Moreno (2003), Moroni (2005), Soler y Lezcano (2009), coinciden en afirmar que el aprendizaje de la programación es complejo, por cuanto implica el manejo de abstracciones, la aplicación de una lógica propia del paradigma de programación y la construcción de expresiones, atendiendo a una sintaxis y a una semántica propias de un lenguaje que no siempre es fácil de asimilar.

Según Oviedo y Ortiz (2002) y Hernández et al. (2006), algunas debilidades de los educandos, que requieren ser tenidas en cuenta por los profesores de Programación, son: el desconocimiento de la materia, pues muchos estudiantes de Ingeniería de Sistemas ingresan al programa sin experiencia previa sobre programación y por ende sin ninguna habilidad relacionada con esta materia; a algunos otros les falta disciplina para aprender a programar, pues el desarrollo de habilidades exige trabajo constante y perseverancia; cuando estas dos situaciones hacen

presencia, puede ocurrir que pierdan el interés por la asignatura, lo que repercute directamente en los resultados que obtienen.

Por su parte, Trejo et al. (2003) consideran que el problema radica principalmente en el desfase que existe entre los temas de Programación, con respecto a los de otras asignaturas del currículo a los que se espera que apoye, como son las Matemáticas y la Física. Una muestra de esta situación es que los conceptos estudiados en los primeros cursos de programación no permiten su aplicación para desarrollar modelos y gráficas matemáticas, ya que esto exige conocimientos más avanzados de programación.

También existe el problema de la diferencia entre el estilo de enseñanza y el estilo de aprendizaje, especialmente cuando no se hace un diagnóstico para determinar los conocimientos previos del estudiante y no se utiliza material educativo apropiado para dar soporte a la enseñanza; además, hace falta una cuidadosa planificación, organización y evaluación del proceso de mejoramiento del curso, como lo mencionan Llamosa et al. (2003), quienes investigaron las dificultades existentes en el proceso de enseñanza-aprendizaje de la programación orientada a objetos.

En la investigación desarrollada por Casas y Vanoli (2007) con el propósito de evaluar los cursos de algoritmos y programación, buscando identificar los obstáculos en el proceso de enseñanza-aprendizaje, se encontró que las actividades que implican mayor dificultad, corresponden al desarrollo de la prueba de escritorio, la identificación de errores y la corrección de los mismos. Este estudio mostró también que los temas que presentan mayor dificultad son la recursividad y el manejo de memoria dinámica.

En este mismo sentido, Pérez (2008) menciona que la Programación es una actividad compleja que requiere: inteligencia, conocimiento, habilidad y disciplina, las cuales sólo son desarrolladas con el paso del tiempo. Para poder programar es necesario poseer: conocimiento del lenguaje, del entorno de programación y dominio de los conceptos y técnicas de programación.

Además de las dificultades anteriores, los autores de este documento consideran que parte del problema en la enseñanza de la programación, es que los ejercicios desarrollados en clase no presentan un nivel de complejidad similar a los problemas reales; por esta razón no permiten desarrollar destrezas en diseño, modelado y programación, como tampoco, aprender todos los conceptos de la asignatura, motivo por el cual, cuando los ingenieros de sistemas tienen que solucionar problemas reales, no cuentan con el conocimiento y la destreza suficientes para hacer un trabajo de calidad.

En consecuencia y con base en De La Cruz y Gamboa (2007), es necesario renovar las estrategias utilizadas para enseñar programación, pues muchas de ellas fueron diseñadas para personas con un perfil diferente al de los programadores actuales, particularmente en cursos de introducción, por la brevedad de los cursos y lo extenso del temario.

Herramientas utilizadas para enseñar Programación

Muchos investigadores, en su afán por mejorar la enseñanza de la programación han desarrollado y experimentado con diferentes recursos educativos; algunos ejemplos de estas investigaciones son:

Satorre et al. (1996) utilizaron el computador y un programa educativo (*Programmin en el laberinto*) para enseñar fundamentos de programación. El *software* permite especificar los algoritmos mediante pseudo-códigos e interactuar y observar la ejecución del algoritmo en una interfaz gráfica 3D. Los autores aseguran que mediante esta estrategia, los estudiantes logran comprender y aplicar conceptos fundamentales como abstracción, modularidad, recursividad y complejidad algorítmica.

En el Departamento de Ciencias e Ingeniería de la Computación de la Universidad Nacional del Sur (Bahía Blanca), Moroni y Señas (1996) desarrollaron un editor para reducir los procesos no cruciales en la resolución de problemas, el diseño y

la formulación de programas, que además permite comprobar errores mediante la confección automática de trazas. Como resultado de esta experiencia, observaron que los estudiantes que utilizaban el editor, necesitaban más tiempo para diseñar los algoritmos que quienes los diseñaban en papel, pero este tiempo estaba compensado por algoritmos más ordenados, con posibilidad de refinamiento progresivo y corrección. El tiempo adicional utilizado en el diseño lo recuperaban en la fase de codificación, actividad que hacían en menor tiempo, debido a que sus algoritmos ya habían sido verificados en el editor y podía generarse el código automáticamente; por otra parte, los estudiantes habían aprendido paulatinamente el lenguaje de programación.

Papert (1999) incursionó en el uso de software educativo como complemento didáctico con enfoque constructivista; además implementó una línea correspondiente a los lenguajes para el aprendizaje, dando como resultado que el lenguaje LOGO, desarrollado en el Instituto Tecnológico de Massachussets, MIT, fue y es utilizado en numerosas escuelas y universidades en un sentido constructivista del aprendizaje, enfoque desde el cual se ha propuesto diferentes estrategias de enseñanza basadas en recursos tecnológicos, como:

- Ambientes de modelación: permiten que el aprendiz construya modelos como LOGO y ambientes basados en éste. (Papert, 1999)
- Ambientes hiper-media: presentan la información en forma no lineal, permitiendo que los estudiantes puedan navegar por ella. El aspecto constructivista está en el hecho de que éstos determinan la secuencia de aprendizaje, pero los conocimientos del dominio en sí, son presentados en forma expositiva. (Bravo, 2000).
- Simulación por computador: son programas que contienen modelos del mundo real, como el juego de roles, el cual según Bergin et al. (2001) es una estrategia de aprendizaje activo en la que se simula escenarios, aplicando los conceptos a enseñar, haciendo que los estudiantes interactúen en ellos, desempeñando el rol que el moderador les asigne. Esta estrategia les permite comprender más fácilmente la información del sistema. La acción básica de éstos, según Hennessy (2003) consiste en realizar cambios en las variables de entrada y observar las consecuencias sobre las variables de salida.
- Estrategias basadas en mapas conceptuales: permiten formar conceptos relacionados entre sí (Cañas y Novak, 2004). Stojanovic (2002) propone su uso para la enseñanza de conceptos básicos de programación y desarrollo de algoritmos. Moreira (2002) por su parte, plantea que los mapas conceptuales desarrollan conexiones con ideas previas, generan capacidad de inclusión, permiten la diferenciación progresiva entre conceptos, e integran o asimilan nuevas relaciones entre ellos.

Además pueden ser usados como herramienta de organización, asociación, validación, interrelación, discriminación, descripción y ejemplificación de contenidos, con un alto poder de visualización (Soler y Lezcano, 2009). Actualmente se cuenta con herramientas que permiten, con gran facilidad, compartir, crear y editar mapas conceptuales, lo que incrementa significativamente su potencial para promover el aprendizaje (Díaz y Leal, 2004). Herramientas como: *Cmap Tool, Mind Mapper, Mind Genius, Concept Draw, Visual Mind, Free Mind y Shared Space*, ofrecen facilidades para el trabajo colaborativo y discusiones de temas, y poseen sofisticadas herramientas de navegación y búsqueda, apertura y creación rápida y fácil de nuevos espacios de información y eficiente ayuda en línea (Copsey, 2005).

- Ambientes colaborativos: buscan propiciar espacios en los cuales se favorezca el desarrollo de habilidades individuales y grupales a partir de la discusión entre los estudiantes al momento de explorar nuevos conceptos. Estos ambientes pretenden que la tecnología apoye el pensamiento creativo, el auto-aprendizaje, el compromiso, la responsabilidad, la participación, la organización, el crecimiento individual

y grupal (Stojanovic, 2002). Lo innovador en ellos es la introducción de la informática, sirviendo las redes virtuales de soporte, lo que da origen a los ambientes *Computer-Support Collaborative Learning* (CSCL) - Aprendizaje Colaborativo Asistido por Computador (Lucero, 2004).

Almeida y sus compañeros (2003) intentaron reducir el vacío creado por la dicotomía entre los estilos de enseñanza y los de aprendizaje, para lo cual desarrollaron una herramienta llamada *EDApplets*, aplicación *web* orientada a la enseñanza/aprendizaje de la programación y de la algorítmica en las ingenierías, orientada principalmente a la animación y visualización mediante trazas de algoritmos y estructuras de datos, que permite descubrir diversos aspectos en una enseñanza, que puede ser dirigida a distintos tipos de aprendizaje: activo/reflexivo, metódico/intuitivo, visual/oral. Los autores de esta investigación no presentan resultados respecto a la contribución de esta herramienta en el aprendizaje, pero se deduce que su uso facilita la enseñanza y permite a los estudiantes disponer de una representación visual del comportamiento de los algoritmos de estructuras de datos, lo que incide favorablemente en la comprensión de los mismos.

Trejo et al. (2003) integraron los temas de programación con aquellos de otras asignaturas, cambiaron el tradicional lenguaje ***C*** por ***Scheme***, que es más sencillo y fácil de aprender, y al mismo tiempo más apropiado para la programación matemática por ser un lenguaje funcional híbrido, obteniendo como resultado que los estudiantes que comenzaron el aprendizaje de la programación con el modelo funcional, mostraron facilidad para comprender y asimilar conceptos avanzados de programación. No obstante, tuvieron que volver a lenguajes más conocidos como Java, dado que pocos docentes conocen Scheme, pero mantuvieron el enfoque integrado al currículo.

Llamosa et al. (2003) presentaron un Sistema Hipermedia Adaptativo para la Enseñanza de los Conceptos Básicos de la Programación Orientada a Objetos, el cual aplica y ajusta la presentación del contenido multimedia conforme al estilo de aprendizaje y nivel de conocimiento alcanzado por un estudiante, para que éste acceda a los contenidos y a las actividades de aprendizaje, de acuerdo con sus características personales. Este estudio permitió a los docentes conocer las tendencias en los estilos de aprendizaje y planificar los cursos de acuerdo con dicho conocimiento. Los estudiantes reciben instrucción más personalizada, lo que incide en su desempeño.

Jiménez et al. (2005) propusieron integrar la visualización y el *role-play* para reducir la complejidad de seguir la traza de los modelos de interacción entre objetos, cuando se genera muchos mensajes. La visualización consiste en representar las características y el comportamiento de los sistemas o de los objetos, utilizando herramientas gráficas, animaciones y otros medios visuales (Price et al. 1993), como es el caso de la representación de las diferentes vistas de un sistema mediante diagramas de Lenguaje Unificado de Modelado, UML. Jiménez et al. (2005) no hacen referencia a la incidencia del *role-play* en el aprendizaje, pero resaltan el aporte a la metodología de enseñanza en la medida en que la herramienta permite mayor posibilidad de interacción del estudiante.

La Fundación Gabriel Piedrahita Uribe ha desarrollado un trabajo de campo sobre la enseñanza de la programación en el Instituto Nuestra Señora de la Asunción, INSA, de Cali y en el Instituto Colombiano de Estudios Superiores de Incolda, ICESI, universidad de la misma ciudad, con el fin de elaborar y validar la herramienta Micromundos y el entorno de programación alterno, *Scratch*, experiencia en la cual participa un grupo de docentes de Informática pertenecientes a tres instituciones educativas. Esta investigación ha mostrado que al resolver problemas mediante programas de computador, los estudiantes comprenden mejor los conceptos matemáticos asociados a los problemas que tratan y mejoran su capacidad para interpretarlos y resolverlos. Con base en los resultados obtenidos

recomiendan que la enseñanza de la programación debe ser incorporada desde la educación primaria (López, 2008).

Pérez (2008) por su parte, propuso una estrategia para mejorar la enseñanza de la programación basada en un ambiente de aprendizaje con un editor interactivo de algoritmos, un constructor automático de trazas y un traductor de pseudocódigo a seis lenguajes de programación. Entre los resultados menciona que diseñar la solución de un problema y generar el código en varios lenguajes mejora el aprendizaje de los estudiantes, y aunque ellos no conozcan previamente esos lenguajes, al ver el código poco a poco lo comprenden y lo aprenden.

También en Pasto se investigó la aplicación del lenguaje *Scheme* como sustituto de los utilizados tradicionalmente (C, Java y *Visual Basic*) en cinco programas de Ingeniería de las instituciones de educación superior. La investigación permitió concluir que el uso de esta herramienta facilita abordar más contenidos debido a su facilidad, y por ser un lenguaje diseñado con base en el modelo de programación funcional, favorece el aprendizaje del manejo de funciones y la aplicación de la recursividad. (Timarán et al, 2009) y (Chaves et al, 2009).

Orozco y Londoño (2009) plantean el problema de que cada vez son menos los estudiantes que ingresan a los programas de Ingeniería de Sistemas, y como solución proponen impulsar un aprendizaje divertido utilizando nuevas herramientas como *Alice y Robot Scribble* en los cursos de programación. En su experiencia y mediante esta estrategia, consiguieron que los estudiantes mejoraran su capacidad algorítmica y la comprensión de los conceptos complejos a través de interacciones divertidas con la interfaz amigable que ofrecen estas herramientas.

Rodríguez et al. (2008) proponen una estrategia consistente en la construcción de programas con interfaz gráfica interactiva, como los videojuegos y los simuladores. El desarrollo de este tipo de aplicaciones permite el estudio de las técnicas de modelado, los componentes de *software* pre-desarrollado y la construcción de un código propio. De igual manera han evidenciado que la motivación de los estudiantes por la asignatura se incrementa notablemente, razón por la cual dedican tiempo adicional entre el 25 y el 100 por ciento del programado en la asignatura, y algunos continúan desarrollando sus aplicaciones después de terminado el curso.

En Pasto, en la Universidad de Nariño, considerando que en la enseñanza de los algoritmos no se aplica la técnica de los diagramas *N-S*, se desarrolló una herramienta denominada ICD-*Chapin*, que permite diseñarlos, probarlos y generar código para el programa en los lenguajes C y Java (Chaves et al, 2008). La primera versión del *software* permite diseñarlos utilizando estructuras secuenciales, selectivas e iterativas. La segunda versión incorpora módulos para el manejo de arreglos y sub-rutinas y fue desarrollada en la Institución Universitaria CESMAG (Cerón y Narváez, 2010). Esta herramienta aún no ha sido evaluada en cuanto a su aporte pedagógico.

CONCLUSIONES

- La programación de computadores es una actividad que en los últimos 50 años pasó de ser un arte, a convertirse en una disciplina con sustento teórico y metodológico. De igual manera dejó de ser una práctica de expertos para convertirse en un conocimiento transversal a muchas disciplinas, como las ingenierías y la informática, siendo así, que algunos investigadores consideran importante su enseñanza desde la educación primaria por considerar que ayuda a desarrollar la lógica para el análisis de problemas y el diseño de soluciones.

- El desarrollo de programas con calidad implica un trabajo interdisciplinario que involucra razonamiento matemático, aplicación de la lógica, diseño y conocimiento de la sintaxis y la semántica de un lenguaje artificial, que son campos de la lingüística y de la inteligencia artificial. Algunas de las temáticas comprendidas en los cursos de programación resultan difíciles de aprender y de aplicar; esto crea la necesidad de planificar cuidadosamente

los cursos y utilizar herramientas que favorezcan la interacción de los estudiantes y la diversidad en las clases para mantener el interés y favorecer el aprendizaje.

- El aprendizaje de la programación es complejo debido a que involucra conceptos abstractos, muchos de los cuales son nuevos para los estudiantes, como: abstracción, modularización, decisiones, iteraciones, recursividad. Por ello, para aprender programación es necesario contar con la dedicación del estudiante, los preconceptos y las herramientas precisas, la disposición del docente, técnicas y estrategias apropiadas para la enseñanza.

- Han sido desarrolladas varias investigaciones basadas en la utilización de diferentes herramientas para apoyar la enseñanza, tanto de diseño de algoritmos como de programación, dentro de las cuales se ha utilizado recursos como: *software* para diseño y evaluación de algoritmos, entornos de programación, juegos, *software* educativo, herramientas visuales y entornos Web. En todos los casos, el uso de herramientas bajo una estrategia de enseñanza ha presentado resultados favorables, ya se trate de despertar el interés de los estudiantes, como de facilitar las interacciones, mejorar la identificación y corrección de errores, o mejorar las habilidades de análisis y solución de problemas. Aunque la mayoría de las investigaciones mencionan resultados positivos en el tratamiento del problema específico al que están orientadas, se observa una limitación en la aplicación de estas estrategias, por cuanto sólo enfrentan un conjunto reducido de síntomas de la problemática asociada con el aprendizaje de la programación.

- Desde la perspectiva curricular se ha encontrado que en muchos programas falta articulación entre las asignaturas de programación y otras asignaturas del Plan de estudios, que deben aportar conceptos y técnicas que ayuden al desarrollo de programas, como son: matemáticas, física, álgebra y estadística. Los ejercicios y proyectos que se desarrolla en las aulas no mantienen relación con el desempeño profesional.

- Se ha desarrollado y probado diferentes herramientas para la enseñanza de la programación y aunque han dado buenos resultados, muchas de ellas no son conocidas y por ello no son aplicadas fuera del contexto donde fueron desarrolladas. Es necesario que los docentes de programación mantengan una actitud investigativa y estén explorando continuamente los avances en el desarrollo de *software* educativo, que puedan incorporar en sus estrategias de enseñanza.

- Muchas estrategias propuestas por los investigadores pueden ser combinadas para conseguir mejores resultados, como por ejemplo: el uso de mapas conceptuales para el estudio de los aspectos teóricos, con herramientas para el diseño y evaluación de algoritmos o programas para el seguimiento de trazas o simulación, según convenga a los temas del curso.

- La programación, al igual que cualquier otra disciplina, no puede ser enseñada a todos los estudiantes y en todos los contextos, por igual. Cada grupo es diferente y requiere una estrategia distinta; por ello es necesario evaluar y renovar las estrategias de enseñanza y los recursos que éstas incluyen. Esto se consigue indagando sobre las investigaciones desarrolladas continuamente en otras instituciones, muchas de las cuales incluyen el desarrollo de nuevas herramientas distribuidas bajo licencia de *software* libre que pueden ser utilizadas sin ningún costo.

- La enseñanza de programación no es un problema aislado, sino que está estrechamente vinculado al currículo del programa y al micro-currículo de la asignatura. Según Julián de Zubiría (1994) el currículo está conformado por seis componentes: propósitos, contenidos, secuenciación, método, recursos y evaluación; de manera que para intentar solucionar un problema de enseñanza habría que considerar todos estos elementos. En este documento se analiza el uso de recursos *software* en la enseñanza de la programación; queda pendiente el estudio de las investigaciones relacionadas con los demás componentes curriculares.

REFERENCIAS BIBLIOGRÁFICAS

Almeida, F., Blanco, V. & Moreno, L. (2003). EDApplets: *Una Herramienta Web para la Enseñanza de Estructuras de datos y Técnicas Algorítmicas*. Recuperado en enero de 2009, en: http://bioinfo.uib.es/~joemiro/aenui/procJenui/ Jen2004/ ponencias/ ponencia48.pdf

Bergin J., Eckstein, J., Wallingford, E. & Manns, M. (2001) *Patterns for Gaining Different Perspectives*. Proceedings of the 8th Conference on Pattern Languages of Programs.

Bravo, J. (2000). *Aprendizaje por descubrimiento en la enseñanza a distancia: Conceptos y un caso de estudio.* Universidad de Castilla La Mancha: Grupo de Informática Educativa, Departamento de Informática

Cañas, A. & Novak, J. (2004) *Concept Maps: Theory, Methodology, Technology*. España.

Casas, S. & Vanoli, V. (2007) *Programación y Algoritmos: Análisis y Evaluación de Cursos Introductorios.* Recuperado en enero de 2009, en: http://www.ing.unp.edu.ar/wicc2007/trabajos/TIAE/156.pdf

Cassola, E. (2004) *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.

Castillo, J. & Barberán, O. (2000) *Mapas Conceptuales en Matemáticas.* Recuperado en octubre de 2009, en: http://www.cip.es/netdidactica/articulos/mapas.htm

Cerón, J. & Narváez, A. (2010) ICD-CHAPIN: I*ntérprete de Comandos para Diagramas Nassi - Shneiderman* (N-S) Versión 2.0. (Trabajo de grado Ingeniería de Sistemas) Institución Universitaria CESMAG, San Juan de Pasto.

Chaves, A., Colunge, C., Ordóñez, H., Checa, J. & Timarán, R. (2009) *Una Experiencia Exitosa en la Enseñanza de Fundamentos de Programación en Ingeniería de Sistemas.* Reunión Nacional y Expoingeniería, ACOFI Septiembre 2009, Santa Marta.

Chaves A., Bastidas, J., Ceballos, E., Timarán, R. & Zamora, P. (2008) *ICD-Chapin: Intérprete de comandos para diagramas N-S*. Primer Congreso Internacional de Gestión Tecnológica e Innovación, Universidad Nacional de Colombia, Bogotá.

Copsey, B. (2005) *Shared Space 2.0 wins TWO REAL Basic Design Awards.* Recuperado en octubre de 2009, en: http://www.shared-space.net/

Dávila, L., González, R., Hernando, M., Platero, C., Rodríguez, D (2008) *Enseñanza de la programación orientada a objetos mediante el desarrollo de aplicaciones gráficas interactivas.* Universidad Politécnica de Madrid, España. Recuperado en julio de 2009, en: http://www3.euitt.upm.es/taee/ Congresosv2/2008/ papers/2008S3B06.pdf

De La Cruz Martínez, G. y Gamboa Rodríguez, F. (2007) *Experiencias con la enseñanza de programación en ambientes colaborativos.* Revista Virtual Educa, Brasil.

De Zubiría, J. (1994) *Los modelos pedagógicos.* Fundación Alberto Merani, Bogotá

Díaz, J., y Leal, P. (2004) *Ambiente Web de Apoyo al Proceso de Enseñanza-Aprendizaje a través de la Representación Gráfica de Significados a modo de Mapas Conceptuales.* Barcelona

Díaz, J. (2006) *Enseñando programación con C++: una propuesta didáctica.* Recuperado en septiembre de 2009, en: http://laboratorios.fi.uba.ar/lie/Revista/Articulos/030307/A2Jun2006.pdf

Fernández Muñoz, L., Peña, R., Nava, F. & Velázquez Iturbide, A. (2002) *Análisis de las propuestas de la enseñanza de la programación orientada a objetos en los primeros cursos.*

JENUI 2002, Jornadas de la Enseñanza de la Informática. Cáceres, España.

Ferreira, A. y Rojo, G. (2006) Enseñanza de la programación. *Revista TE&ET Iberoamericana de Tecnología en Educación y Educación en Tecnología* (En Línea), Universidad Nacional de Río Cuarto, Argentina, Vol. 1, Nº 1, diciembre. Recuperado de: http://teyet-revista.info.unlp.edu.ar/files/No1/09_ Enseñanza_de_la_programación.pdf

García, J. (2003) *Un Enfoque semi-formal para la Introducción a la Programación*. Departamento de Informática y Sistemas, Universidad de Murcia, España.

Gayo, D., Cernuda, A., Cueva, J., Díaz, M., Almudema, M. & Redondo, J. (2003) *Reflexiones y experiencias sobre la enseñanza de POO como único paradigma.* Recuperado en septiembre de 2009, en: http://www.di.uniovi.es/~dani/publications/ jenui03.pdf, Universidad de Oviedo.

Hennessy, S. (2003) *Learner perceptions of realism and "magic" in computer simulations.* British Journal of Educational Technology, Nº 24.JEITICS 2005 - Primeras Jornadas de Educación en Informática y TIC en Argentina, Laboratorio de Investigación y Desarrollo en Informática y Educación (LIDInE), Instituto de Investigación en Ciencias y Tecnología Informática (IICTI), Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca.

Jiménez-Díaz G., Gómez-Albarrán M., Gómez Martín M.A. & González Calero P.A. (2005) *Visualización y Role-Play en la Enseñanza de la Programación Orientada a Objetos.*

VII Simposio Internacional de Informática Educativa – SIIE05, Leiria, Portugal, 16-18 de noviembre

Kölling, M. (1999) The *Problem of Teaching Object-Oriented Programming,* Part 1: Languages in Journal of Object-Oriented Programming, Vol. 11, Nº 8

Levy, L. (1994) *From Specific Problem Instances to Algorithms in the introductory course.* Sigcse Bulletin Acm

Llamosa R., Guarín I., Moreno G. & Baldiris S. (s/f) S*istema Hipermedia Adaptativo Para la enseñanza de los Conceptos Básicos de la Programación Orientada a Objetos.* Recuperado en agosto de 2009, en:http://lsm.dei.uc.pt/ribie/docfiles/txt2003326195840A016.pdf

Londoño, G. & Paz, G. (2007) *Programación Básica para Adolescentes*. Revista Sistemas y Telemática, ICESI, Cali.

López García, J. (2008) *Algoritmos y Programación en la Educación Escolar.* Recuperado en enero de 2009. En: http://www.eduteka.org/ pdfdir/FGPUPonenciaAlgoritmos.pdf

Lucero, M. (2004) Entre el trabajo colaborativo y el aprendizaje colaborativo. *Revista Iberoamericana de Educación,* Nº 5-9

Mc Carthy, J. (1965) Lisp: Programming Manual. Cambridge.

Moreira, M. A. (2002) *Mapas conceptuales y aprendizaje significativo.* Recuperado en octubre de 2009, en: http://www.if.ufrgs.br/~moreira/mapasesp.pdf

Moroni-Perlas, N. (2005) *Estrategias para la enseñanza de la programación*. JEITICS 2005 - Primeras Jornadas de Educación en Informática y TICS en Argentina.

Moroni-Perlas, N. (1996) *Un entorno para el aprendizaje de la programación.* (CACIC incompleto). Recuperado en Octubre 2009, en: www.mendeley.com/research/redes-colaborativas-tecnologa-e-identidades-un-acercamiento...

Orozco, A. & Londoño, S. (2009) *Nuevas Herramientas y Metodologías en la Enseñanza de la Programación Usando Alice y Robot Scribble.* Reunión Nacional y Expoingeniería ACOFI, Septiembre 2009, Santa Marta.

Oviedo, M. y Ortiz, F. (2002) *Enseñanza de la programación.* Academias de Computación de la UPIICSA, Institución: IPN-UPIICSA, México D.F.

Papert, S. (1999) *¿Qué es Logo? ¿Quién lo necesita?* Logo Philosophy and Implementation. LCSI Review.

Perez, R. (2008) U*na Herramienta y Técnica para la Enseñanza de la Programación.* Recuperado en agosto de 2009, en: http://campusv.uaem.mx/cicos/imagenes/memorias/6tocicos2008/Articulos/Cartel%206.pdf

Pérez Y. & López L. (2007) *Multi-paradigma en la Enseñanza de la Programación.* Universidad Nacional de Comahue, Buenos Aires. Recuperado en octubre de 2009, en: http://www.ing.unp. edu.ar/ wicc2007/trabajos/TIAE/153.pdf

Price B.A., Baecker R.M. & Small I.S. A. (1993) Principled Taxonomy of Software Visualization in *Journal of Visual Languages and Computing*, Nº 4, 3.

Redondo J., Ortín F. & Vinuesa L. (2005) *Aplicación de Técnicas de Evaluación para la enseñanza de la programación.* I Jornada de Innovación docente de la EUITIO, Oviedo, 19 y 20 de octubre 2005.

Satorre, R., Llorens, F. & Puchol, J. (1996) *Enseñar programación en las Ingenierías Informáticas*. II Jornadas Nacionales de Innovación en las Enseñanzas de las Ingenierías, Instituto de Ciencias de la Educación, Universidad Politécnica de Madrid.

Soler, Y. y Lezcano, M. (2009) Consideraciones sobre la tecnología educativa en el proceso de enseñanza-aprendizaje. Una experiencia en la asignatura Estructura de Datos. *Revista Iberoamericana de Educación*, Nº 49/2 – 10 de abril de 2009, EDITA: Organización de Estados Iberoamericanos para la Educación, la Ciencia y la Cultura (OEI).

Steele, JR. (2006) Objects have not failed. Recuperado en septiembre de 2009, en: http://www.dreamsongs.com/ObjectsHaveNotFailedNarr.html

Stojanovic, L. (2002) El paradigma constructivista en el diseño de actividades y productos informáticos para ambientes de aprendizaje. *Revista Pedagogía,* Caracas, Vol. 23, Nº 66.

Timarán, R., Chaves, A., Checa, J., Ordóñez, H., Colunge, C. & Jiménez, J. (2009a) Un Cambio de Paradigma en la Enseñanza de Fundamentos de Programación en Ingeniería de Sistemas. *Revista Educación en Ingeniería,* junio 2009.

Timarán, R., Chaves, A., Checa, Colunge, C., J., Jiménez, J. & Ordóñez, H (2009b) *Un nuevo enfoque en la enseñanza de la programación.* Universidad de Nariño, San Juan de Pasto.

Trejo, R., Santiago, R., Quezada, L. & Delgado F. (2003) La Enseñanza de la Programación Dentro de un Modelo de Integración Curricular: la Experiencia del Proyecto Principia. Recuperado en septiembre de 2009, en: http://lsm.dei.uc.pt/ ribie/docfiles/txt20031212172724TCI15.pdf

Villalobos, J. & Casallas, R. (2006) F*undamentos de programación: aprendizaje basado en casos.* México: Prentice Hall.

Villalobos, J., Casallas, R. & Marcos, C. (2007) El Reto de Diseñar un Primer Curso de Programación de Computadores. Recuperado en octubre de 2009, en: http://cupi2.uniandes.edu.co/ docs/Villalobos-Casallas-Marcos-CIESC05.PDF

Wirth, N. (1989) *Algoritmos e Estructuras de Dado*. LTC Informática-Programação, México D.F.: Grupo Macmillan, Ediciones El Castillo.

Zhu, H. & Zhou, M. (2003) *Methodology First and Language Second: A Way to Teach Object-Oriented Programming* OOPSLA'03, Anaheim, CA.