MATICES Tecnológicos
Revista de la Facultad de Ciencias Naturales e Ingeniería

LA DOCENCIA ESTRATÉGICA COMO PROPUESTA PARA ENSEÑAR PROGRAMACIÓN

Chaves Torres, Anívar[1]
Escuela de Ciencias Básicas, Tecnología e Ingeniería
Universidad Nacional Abierta y a Distancia UNAD

anivarchaves@yahoo.com

[1] Ingeniero de sistemas. Especialista en Docencia Universitaria y Magíster en Educación. Docente de la Universidad Nacional Abierta y a Distancia. Se ha desempeñado en varias empresas e instituciones de educación superior, y como desarrollador de software independiente.

Resumen— La enseñanza de la programación de computadores tiene un papel relevante en la formación de ingenieros de sistemas y otros profesionales afines. Aprender a programar implica el dominio de un amplio marco conceptual, el conocimiento de una serie de herramientas indispensables para la construcción de software y el desarrollo de habilidades de abstracción, diseño, modelado, comunicación y verificación. Todo esto, orientado desde una estrategia de solución de problemas, debe posibilitar la implementación de productos software de alta calidad.

Diversas investigaciones se han enfocado a identificar las dificultades existentes en la enseñanza y el aprendizaje de la programación, así como a proponer diferentes tipos de soluciones, basándose en algunos factores como: paradigma, enfoque y herramientas. En este artículo se hace una revisión de algunos de los aportes más relevantes sobre el tema y se presenta una propuesta metodológica de docencia estratégica.

Palabras clave— enseñanza, programación, enfoque, metodología

Abstract **-** The computer programming teaching has a relevant paper on the systems engineers and other related professionals formation. To The learning to programming implicates to dominate a ample conceptual frame, to know a several indispensable tools for build software and to develop abilities of abstraction, design, model, communication, and verification. All this oriented from a strategy of problems solution must enable to construct software of high quality.

Many investigations have tried to identify the difficulties about programming teaching and learning and to propose different types solutions with base on some factors as: paradigm, focus and tools. This article presents a reviewing of some more relevant contributions about this theme and includes a methodological proposal of strategic teaching.

Keywords - teaching, programming, focusing, methodology

I. INTRODUCCIÓN

Partiendo del planteamiento de Wirth [1]: "La programación es el arte y la técnica de construir y formular algoritmos de una forma sistemática", la enseñanza de la programación debe orientarse a capacitar al estudiante para solucionar problemas de todo tipo y de manera organizada, como sugiere Moursund [2]; o, como afirma López [3], para elaborar estructuras mentales para la solución de problemas.

Tal importancia tiene la programación en la formación profesional actual, pues es una de las áreas disciplinares en los currículos de sistemas, computación e informática. Muchos programas incluyen varios cursos relacionados con programación, tanto en los primeros semestres como en los últimos. Esto, sumado al profuso número de programas ofrecidos por las instituciones de educación superior, crea una alta expectativa sobre la producción de software en Colombia que, desafortunadamente, no corresponde a la realidad.

Aquí se evidencia la brecha existente entre enseñar y aprender, pues el hecho de que se enseñe mucha programación a muchos estudiantes no garantiza que haya suficientes y competentes desarrolladores de software. A la vista del neófito parecería que la enseñanza de la programación es tarea fácil, pero la experiencia muestra una perspectiva diferente. Prueba de ello es que en los últimos diez años se ha realizado un gran número de investigaciones sobre el tema y se han propuesto diferentes enfoques, estrategias y herramientas como se puede apreciar en los aportes de Cassola [4], Kölling [5], Zhu y Zhou [6], Villalobos y Casallas [7] y García [8]. Sin embargo, no parece existir aún una solución completamente satisfactoria y generalizable.

Un indicador de la problemática en relación con la enseñanza de la programación es el alto índice de reprobación de asignaturas de esta área. Según Hernández y sus colaboradores [9], las principales dificultades en el aprendizaje de la programación se pueden resumir en:

- Exigencia de un elevado nivel de abstracción
- Necesidad de conocimientos previos sobre temas básicos y manejo de técnicas de resolución de problemas.
- Heterogeneidad en los conocimientos y habilidades de los estudiantes, lo que se traduce en ritmos de aprendizaje muy variados y tal vez descompensados.
- Grupos numerosos, lo que impide el desarrollo de un trabajo individualizado por parte del docente, que tenga en cuenta las características particulares.
- La imposibilidad de asumir un único método para la resolución de los problemas.

En este artículo se analizan algunos de los aportes identificados desde la perspectiva del paradigma, el enfoque y las herramientas utilizadas; se esboza una propuesta metodológica y finalmente se presentan las conclusiones sobre el tema.

II. ENSEÑANZA DE LA PROGRAMACIÓN

Este tema puede ser abordado desde múltiples perspectivas, pero por razones de tiempo, esfuerzo y espacio, en este análisis se abordan únicamente cuatro: el paradigma de programación, el enfoque, las herramientas y la metodología.

A. Paradigmas de programación

Aunque existen muchos paradigmas, con sus conceptos, técnicas y herramientas, la enseñanza de la programación ha privilegiado a un número reducido de ellos: programación estructurada, programación orientada a objetos y, en menor grado, programación funcional. Otros paradigmas como: programación lógica, programación por restricciones y programación multiparadigma, sólo se abordan para determinadas asignaturas o en cursos electivos.

Los paradigmas de programación han surgido en diferentes momentos y para atender necesidades propias de dicho momento. García [8] considera que se han presentado tres etapas en la evolución de la programación: en la primera, la programación es un arte donde lo más importante es el talento del programador y la potencia del lenguaje; en la segunda, se reconoce la necesidad de darle un énfasis más científico y se consideran propuestas como: programación estructurada, refinamiento por pasos sucesivos, tipos de datos, abstracción de datos, ocultación de información y jerarquía de tipos; es la etapa en que tiene auge la programación estructurada. Finalmente, en la tercera, se centra el interés en el modelo orientado a objetos, por ser el más adecuado para producir software de calidad y de gran tamaño; se hace énfasis en el modelado, la definición de estándares y el uso de patrones. Este modelo supera al estructurado al permitir la reutilización de código mediante el concepto de herencia y polimorfismo y mejora la abstracción mediante el concepto de encapsulado.

Mientras en la primera etapa la estrategia más importante para los programadores consistía en la descomposición procedural de los programas; y en la segunda, en el diseño de algoritmos eficientes; en la tercera, desde el modelo orientado a objetos, se basa en el modelado conceptual.

Al tiempo que tomaba fuerza la programación estructurada, otros importantes investigadores [10] trabajaban en la aplicación de conceptos matemáticos al campo de la programación, especialmente lo relacionado con funciones, dando como resultado el paradigma de programación funcional.

En el ámbito universitario, la opinión sobre el paradigma con el que se debe enseñar programación está dividida entre los modelos: estructurado, orientado a objetos y funcional.

Hay quienes consideran que lo importante en los fundamentos de programación son los algoritmos y su diseño, aplicando las estructuras de programación para posteriormente ser implementados en un lenguaje de programación estructurado.

Entre estos se encuentran Ferreira y Rojo [11]. También hay quienes recomiendan enseñar primero el modelo estructurado y luego el orientado a objetos, porque consideran que los conceptos de éste último son más avanzados y se comprenden con conocimientos básicos provenientes del modelo estructurado.

Respecto al modelo orientado a objetos, actualmente hay un número creciente de docentes e investigadores que se han inclinado por la enseñanza de la programación bajo este paradigma, como Gayo [12]; considerando que éste permite hacer mejores abstracciones y construir fácilmente modelos de los sistemas reales, como también, que es el modelo más utilizado para la construcción de software en todo el mundo.

Por otra parte, también se ha investigado la aplicabilidad del modelo funcional para comenzar la enseñanza de la programación. Con base en su investigación, Chaves [13] propone enseñar los fundamentos de programación desde el modelo funcional utilizando un leguaje como Scheme. A la vez que el modelo permite el razonamiento matemático para diseñar soluciones, el lenguaje facilita su codificación. De esta manera, el estudiante mejora la comprensión de los conceptos de programación como la recursividad y la habilidad para proponer soluciones.

En este mismo sentido, Timarán[14], después de haber comparado los resultados en cuanto a conocimientos teóricos y habilidad de solución de problemas en estudiantes que cursaron asignaturas de programación utilizando lenguajes imperativos como C y Java y los que abordaron el modelo declarativo utilizando lenguaje Scheme, recomienda cambiar del paradigma imperativo al declarativo.

Finalmente, hay también quienes consideran que es necesario abordar la enseñanza desde varios paradigmas. Entre ellos Pérez y López [15], quienes consideran que los paradigmas de programación, con sus técnicas y herramientas, son aplicables a dominios particulares de problemas.

Los estudiantes que aprenden a programar en el marco de un paradigma pretenden solucionar todos los problemas desde el paradigma que conocen, sin preguntarse si habrá otra forma y otras herramientas más adecuadas para el problema en particular. En consecuencia, recomiendan enseñar la programación abordando el mayor número de paradigmas desde los primeros cursos, haciendo uso de lenguajes multiparadigma.

B. Enfoque

Los primeros programas eran el resultado de un proceso creativo más que de un razonamiento riguroso sobre un diseño de solución, dada la precariedad de conocimientos sobre el tema; lo importante era que el programa se ejecutara y desarrollara las operaciones para las que fue construido. Dijkstra y Feijen [16] criticaron esta forma de programar, considerándola artesanal, e insistieron en que la corrección del programa debería demostrarse mediante un proceso formal de razonamiento a partir de su especificación. En este orden de ideas, lo importante para el estudiante de programación no es escribir el código del programa y hacer que se ejecute, sino dar prueba de que el algoritmo que diseña para solucionar el problema es correcto y eficiente.

Van [17] consideró tres enfoques posibles en la enseñanza de la programación: formal, semiformal y no formal. El enfoque formal propende por una aplicación de fundamentos matemáticos para diseñar algoritmos en los que se pueda demostrar formalmente su corrección. Este enfoque es apropiado para los matemáticos y para estudiantes de ciencias de la computación.

El enfoque semiformal busca la corrección con base en una derivación formal, pero admite cierto margen de informalidad considerando que no es posible demostrar matemáticamente la corrección en la solución de todos los problemas. Este enfoque es apropiado para ingenieros de sistemas e informáticos.

Finalmente, el enfoque no formal no exige el razonamiento matemático, de manera que los programas pueden construirse mediante prueba y error, intentando una y otra vez hasta lograr una solución aceptable. Este enfoque es admisible en los niveles de formación técnica y en programas profesionales donde la programación es un área complementaria.

C. Herramientas

Para escribir un programa, dependiendo del modelo y del lenguaje a utilizar, se requieren herramientas como: máquina virtual (MV), compilador o intérprete, entorno integrado de desarrolla (EDI), marcos de trabajo (FW), interfaces para la programación de aplicaciones (API) o Librerías.

Algunas herramientas están diseñadas para satisfacer las necesidades básicas e indispensables en la programación y son fundamentales para la enseñanza. Otras, por el contrario, son herramientas de cuarta generación capaces de automatizar gran parte del trabajo del programador; éstas pueden escribir código a partir de la especificación de un algoritmo o de un modelo de diseño, y fueron diseñadas para la producción de software y no para la enseñanza.

El hecho de que una aplicación ofrezca buenos resultados en la industria no es garantía de que también los ofrezca en la enseñanza, dado que los objetivos son diferentes. Las herramientas deben permitirle al estudiante poner en práctica las nociones de programación y la experimentación, pero no hacer el trabajo por él ya que le suprimirían la oportunidad de aprender.

Muchos investigadores se han dado a la tarea de desarrollar herramientas didácticas para apoyar la enseñanza de la programación; algunos ejemplos relevantes se presentan a continuación.

Blanco y Moreno [18] presentan EDApplets, una aplicación web orientada a la enseñanza/aprendizaje de la programación y de la algorítmica en ingeniería, orientada principalmente a la animación y visualización mediante trazas de algoritmos y estructuras de datos, que permite descubrir diversos aspectos en la enseñanza y puede ser dirigida a distintos tipos de aprendizaje: activo/reflexivo, metódico/intuitivo, visual/oral.

Llamosa [19] y su equipo proponen un Sistema Hipermedia Adaptativo para la Enseñanza de los Conceptos Básicos de la Programación Orientada a Objetos. Este herramienta, según sus autores, fue diseñada para identificar el estilo de aprendizaje y el nivel de conocimiento de cada estudiante, adaptando la presentación de los contenidos para que cada quien acceda a ellos y a las actividades de aprendizaje, de acuerdo con sus características personales.

La Fundación Gabriel Piedrahita Uribe desarrolló un trabajo de campo sobre la enseñanza de la programación en el Instituto Nuestra Señora de la Asunción (INSA) de Cali y en la Universidad ICESI 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. Juan Carlos López [20], con base en los resultados obtenidos, recomienda que la enseñanza de la programación sea incorporada desde la primaria.

También se conoce una estrategia propuesta por Pérez [21] para mejorar el aprendizaje de la programación basada en un ambiente de aprendizaje con un editor interactivo de algoritmos, un constructor automático de trazas y un traductor a programas en lenguaje Pascal.

Los programas Alice y Robot Scribble fueron utilizados por Orozco y Londoño [22] para promover un aprendizaje divertido y consiguieron que los estudiantes mejoraran su capacidad algorítmica y la comprensión de los conceptos complejos a través de interacciones con la interfaz amigable que ofrecen estas herramientas.

También se han llevado a cabo investigaciones para validar la utilización de lenguajes de programación no convencionales, como menciona Trejo [23]. Estos investigadores integraron los temas de programación con los de otras asignaturas, y cambiaron el tradicional lenguaje C por Scheme, que es un lenguaje funcional híbrido, más sencillo y fácil de aprender, y al mismo tiempo más apropiado para la programación matemática. No obstante los resultados positivos, tuvieron que volver a lenguajes más conocidos como Java, dado que pocos docentes conocen Scheme, pero mantuvieron el enfoque integrado al currículo.

Un trabajo similar fue desarrollado 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 instituciones de educación superior. La investigación permitió concluir que el uso de esta herramienta facilita abordar más contenidos debido a que se requiere menos tiempo para aprender el lenguaje y, por ser un lenguaje funcional, favorece el aprendizaje del manejo de funciones y la aplicación de la recursividad como lo corroboran Chaves [13] y Timarán [14].

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 diseñar los algoritmos, probarlos y generar código en los lenguajes C y Java, desarrollado por Chaves [24]. La primera versión del software permite diseñar algoritmos utilizando estructuras secuenciales, selectivas e iterativas. La segunda versión incorpora módulos para el manejo de arreglos y sub-rutinas y fue desarrollada por Cerón y Narváez [25], en la Institución Universitaria CESMAG.

Por otra parte, en relación con el uso de herramientas tanto para enseñar como para programar, está el problema de las licencias del software. Aunque existe una amplia gama de licencias, para este caso se consideran únicamente dos: las herramientas cuyos derechos pertenecen a una organización y que su uso implica un contrato de licenciamiento (uso privativo) y las que cuentan con licencia de propósito general (GPL), comúnmente conocidas como software libre.

Ciertamente que algunas instituciones pueden permitirse la adquisición de licencias educativas para diferentes plataformas y herramientas de desarrollo; también ocurre que las empresas productoras de software ofrecen sus productos a las instituciones a bajo costo o de forma gratuita con el fin de hacerlos conocer por los futuros profesionales como una estrategia de mercadeo a largo plazo. Es preciso tener en cuenta que, a menos que se estén formando empleados para un determinado sector empresarial donde las plataformas ya están establecidas y los egresados se convierten en usuarios finales, es conveniente ofrecerles herramientas que ellos puedan utilizar tanto dentro como fuera de la universidad, con fines educativos y productivos.

Como afirma Stallman, sólo el software que se distribuye bajo licencia GPL ofrece a los usuarios "la libertad para ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software" [26]. En el ámbito universitario, la copia y la distribución son comunes ya se trate de software libre o privativo, pero únicamente el software libre contribuye directamente al aprendizaje de los estudiantes al posibilitar el estudio de la arquitectura y el código de las propias herramientas y su modificación para adaptarlas a las necesidades y preferencias de quien las estudia, llegando incluso a generar nuevas versiones del software, las que se pueden distribuir libremente.

Hay razones generales por las que todos los usuarios de computadoras deberían usar software libre, como es la libertad de controlar sus propios equipos, pues el uso de software privativo lo obliga a depender del proveedor en la medida en que el sistema ejecuta órdenes prediseñadas por el

propietario del software. El software libre también da a los usuarios la posibilidad de cooperar unos con otros a través de los proyectos de desarrollo colaborativo en los que pueden participar con diferentes roles, tal como lo argumenta Stallman [27]. Adicionalmente, las instituciones educativas encuentran que el software libre supone un ahorro de costos, además de libertad de copiar y redistribuir los programas en cualquier cantidad de computadores. Esto no es lo mismo que conseguir una licencia gratis de una empresa productora de software donde transcurrido un tiempo se deberá pagar por actualizaciones.

D. Metodología

El ser humano está capacitado para llevar a cabo aprendizajes complejos, pero el éxito de los mismos dependerá de las posibilidades que le ofrezca el medio cultural a través de diversas herramientas o amplificadores de las capacidades: motoras, sensoriales y racionales. En este sentido, el aprendizaje de la programación no depende solo de la disciplina, disposición y perseverancia de cada estudiante, sino también de las competencias pedagógicas del profesor y de las condiciones que el medio le ofrezca.

El conocimiento en el campo pedagógico y didáctico ha avanzado al igual que en el campo de la programación; no obstante, en muchos casos, la metodología con la que los docentes desarrollan los cursos no ha cambiado en varios años y por ende no se adapta a las circunstancias actuales. Hoy en día, la mayoría de los estudiantes disponen de equipos de cómputo y conexión a Internet, lo cual facilita el acceso en línea a recursos educativos como: libros, manuales, tutoriales, apuntes, problemas, experimentos, diapositivas, aplicaciones de exámenes virtuales y todo tipo de software. El trabajo del docente no puede limitarse a transferir información cuando la misma está disponible en todo lugar y a toda hora. En este punto surge una pregunta fundamental: ¿Cómo enseñar programación?

III. DOCENCIA ESTRATÉGICA

En atención a las necesidades actuales en el campo de la programación y considerando el estudio del tema presentado hasta el momento, se propone asumir la enseñanza de la programación desde el concepto de docencia estratégica. Este enfoque establece que la docencia debe buscar el aprendizaje significativo de los contenidos y el desarrollo de habilidades de pensamiento para que los estudiantes se conviertan en aprendices autosuficientes. Para lo cual es necesario el desarrollo de las cuatro acciones propuestas por Tobón [28]: diagnóstico, planeación, valoración y monitoreo.

A. Diagnóstico

Esta actividad provee al docente del conocimiento suficiente sobre el grupo, sobre el curso a desarrollar y sobre las circunstancias en las que se va a llevar a cabo. Este conocimiento es fundamental para realizar la planeación.

En el diagnóstico se recomienda comenzar indagando sobre los conocimientos previos de los estudiantes, los estilos de aprendizaje, la experiencia en temas relacionados con el curso y las características relevantes del grupo. Con base en dicho conocimiento se podrá determinar los propósitos de formación para el curso en particular.

De igual manera, es importante que el docente se informe sobre recursos educativos digitales disponibles. Cada año se desarrollan nuevos programas, algunos de gran valor didáctico, y muchos no son aprovechados por desconocimiento de los docentes. Un error frecuente que estos comenten es creer que se puede desarrollar el mismo curso con diferentes grupos y que si una vez fue exitoso lo seguirá siendo; cada grupo de estudiantes es diferente, por eso el docente debe estudiarlo y diseñar su curso para cada grupo en particular. Aunque la temática sea la misma, la metodología ha de ser diferente.

B. Planeación

Esta actividad proporciona un diseño detallado del curso a desarrollar teniendo en cuenta los propósitos de formación, los recursos educativos y locativos disponibles, el tiempo y las características particulares del grupo. Algunos aspectos importantes a tener en cuenta en un curso de programación son:

- El tema, la estrategia, las actividades y los recursos deben despertar en el estudiante el deseo de aprender desde la primera sesión y mantenerse durante todo el curso. Crear el ambiente apropiado para el aprendizaje es una tarea que el docente debe tener presente desde el momento de la planeación y estar pendiente de ello durante las sesiones de clase.

- La programación exige conocimiento y habilidad mental para la solución de problemas. Alcanzar dicho nivel requiere dedicación y disciplina por parte de los estudiantes. En atención a esto, la estrategia del docente debe permitir presentar cada tema de forma que resulte fácilmente comprensible y asimilable para el estudiante; para ello es importante saber cuáles son los conocimientos previos de los estudiantes, de manera que los temas se presenten como continuación de lo que ellos ya conocen.

- Algunos docentes comienzan el curso intimidando a sus estudiantes, diciéndoles que su asignatura es muy difícil, pensando que por ello dedicarán más interés y tiempo a estudiarla; pero esto funciona justamente en sentido contrario. Un estudiante se interesa en aprender lo que considera importante y de su agrado, pero posterga lo que considera difícil; es más, presenta resistencia a aprender lo que considera complicado; por ello, el docente ha de resaltar la importancia del tema, pero presentarlo de tal forma que resulte comprensible, interesante y agradable. No se trata únicamente de presentar a los estudiantes los

contenidos del curso, sino de hacer de ellos unos amantes y estudiosos de la programación. No hay necesidad de anunciar las dificultades del curso; si estas se presentan los estudiantes serán quienes más directamente las experimentarán y aprenderán de ellas.

- El equilibrio entre la teoría y la práctica es fundamental en la enseñanza de la programación, pues favorecer un aspecto en detrimento del otro resulta perjudicial en cualquier caso. Si se da mayor valor a la teoría, el estudiante tendrá problemas al pasar a la práctica y perderá el interés por los conceptos estudiados; si se privilegia la práctica, el estudiante menospreciará la teoría e intentará resolver todos los problemas con un mínimo de conceptos desarrollando un estilo artesanal de programación. La teoría y la práctica deben articularse mediante casos de estudio desarrollados de manera completa y sistemática, donde la teoría guía el proceso y explica las acciones, mientras que la práctica da cuenta de cómo solucionar el problema haciendo uso de la teoría.

C. Valoración

Esta actividad se lleva a cabo aplicando criterios coherentes con los propósitos de formación y analizando las evidencias de los logros alcanzados.

En esta valoración se debe tener en cuenta el progreso de los estudiantes en la asimilación de conceptos y en la habilidad desarrollada para la solución de problemas; es decir, en la capacidad para diseñar una solución fundamentada en la teoría y llevarla a la práctica cumpliendo unas condiciones de calidad.

De igual manera se debe considerar la disposición del docente y de los estudiantes hacia la enseñanza y el aprendizaje, la pertinencia de las actividades y los recursos utilizados, así como el impacto de la estrategia.

D. Monitoreo

Esta actividad se desarrolla de manera permanente y tiene como propósito conocer la efectividad de la estrategia y hacer las modificaciones necesarias en el momento oportuno. Consiste en cotejar la planeación con la valoración para identificar la pertinencia de la estrategia para el logro de los propósitos de formación y hacer los ajustes necesarios en caso de identificarse una desviación en los resultados.

IV. CONCLUSIONES

Un curso de programación debe conseguir que los estudiantes asimilen los conceptos propios del modelo de programación; desarrollen la habilidad de aplicarlos para interpretar, modelar, diseñar e implementar soluciones a problemas del mundo real; y adquieran la destreza en el manejo de un lenguaje de programación y sus herramientas para escribir programas correctos y eficientes.

Entre los paradigmas mencionados: estructurado, orientado a objetos y funcional, el más adecuado para la enseñanza de la programación es el orientado a objetos. El modelo estructurado es más sencillo de enseñar, dado que aplica una cantidad menor de conceptos y son más fáciles de comprender que los del orientado a objetos, pero su utilización en el desarrollo de software profesional está disminuyendo rápidamente.

Por su parte, el modelo funcional, aunque genera programas más pequeños, exige la aplicación de razonamiento matemático que generalmente causa dificultad a los estudiantes, además de que es aplicable a un conjunto más reducido de problemas.

Finalmente, el modelo orientado a objetos, por su parte, aplica la mayoría de los conceptos del modelo estructurado al interior de los métodos, pero va mucho más allá al incluir los conceptos de tipificación, jerarquía y encapsulado; cuenta con una amplia documentación tanto en español como en inglés y con multiplicidad de recursos bajo licencia GPL, además de ser el modelo más utilizado en todo el mundo.

Respecto al nivel de formalidad aplicado en el diseño de las soluciones, se considera que el enfoque formal debe ser la meta, pero es muy difícil de lograr en la formación de pregrado. Tampoco se puede pasar al otro extremo de permitir que los estudiantes asuman como método para programar la técnica iterativa: escribir-probar-corregir, sin un diseño previo, sin una evaluación de la corrección sobre el modelo de diseño. En consecuencia, se propone aplicar el nivel semiformal. Esto implica que para cada programa se llevará a cabo un procedimiento sistemático que incluya análisis y modelado de requisitos, la construcción de un diseño en el que se pueda rastrear los requisitos y la verificación de su corrección antes de proceder a escribir el código.

Para facilitar la experimentación y el aprendizaje es necesario que tanto las instituciones como los mismos estudiantes cuenten con las herramientas apropiadas y suficientes. Aunque las universidades cuenten con licencias para utilizar herramientas de tipo propietario, es necesario que los docentes fomenten el conocimiento y utilización de herramientas con licencia GPL. El software libre facilita la experimentación, la investigación y estimula la creatividad; además de que brinda a los estudiantes la oportunidad de pertenecer a nuevos grupos académicos con reconocimiento y diversos estímulos, acercándolos al desarrollo de la tecnología.

Desde lo pedagógico, la docencia estratégica ofrece los lineamientos para el diseño y desarrollo de los cursos de programación, fomentando en los estudiantes el aprendizaje autónomo y el desarrollo de habilidades que les permitan un buen desempeño profesional.

Entonces, la función docente no se limita a la transferencia de saberes; por el contrario, es un compromiso que trasciende la relación con los estudiantes y con las instituciones y se

proyecta a la sociedad, en un mediano y largo plazo. Por esto, es importante realizar un trabajo cuidadoso en el que se evidencie la intencionalidad y el nivel de compromiso a través de la aplicación de una estrategia como la que se presenta en este documento, en la que se tengan en cuenta los conocimientos previos de los estudiantes y sus características y circunstancias particulares, se lleve a cabo la planeación del curso con base en dicha información, se evalúe de acuerdo a lo planeado, apoyándose en una observación objetiva y constante del acontecer en el aula, y adicionalmente se tenga la voluntad de hacer los ajustes que se consideren pertinentes.

La docencia estratégica es un reto que va más allá del compromiso laboral, pero los docentes que se decidan a llevarla a cabo tendrán como recompensa la satisfacción de contar estudiantes mejor cualificados para el desempeño profesional.

Referencias

[1] N. Wirth, "Program Development by Stepwise Refinement". Communications of the ACM, Vol. 14, No. 4, April. 221 – 227, 1971.

[2] D. Moursund, "Computational Thinking and Math Maturity: Improving Math Education" [En Línea]. K-8 Schools. 2008. Disponible en: http://uoregon.edu/~moursund/Books/ElMath/ElMath.html

[3] J. López. Algoritmos y Programación en la Educación Escolar. [En línea]. 2008. Disponible en: http://www.eduteka.org/pdfdir/FGPUPonenciaAlgoritmos.pdf

[4] E. Cassola. 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.

[5] M. Kölling. The Problem of Teaching Object-Oriented Programming, Part 1: Languages. Journal of Object-Oriented Programming, vol. 11, num 8, 8-15, 1999.

[6] H- Zhu and M. Zhou. Methodology First and Language Second: A Way to Teach Object-Oriented Programming. OOPSLA'03, Anaheim, CA, 2003.

[7] J. Villalobos and R. Casallas. Fundamentos de programación: aprendizaje basado en casos. Prentice Hall, México. 2006.

[8] J. García. Un Enfoque Semiformal para la Introducción a la Programación. Departamento de Informática y Sistemas. Universidad de Murcia, España. 2003.

[9] Hernández, et al. Utilización de estructuras de aprendizajes IMS-LD en la enseñanza de la programación. Current Developments in Technology-Assisted Education, 2006.

[10] J. Mccarthy. Lisp: Programming Manual. Cambridge. 1965.

[11] A. Ferreira y G. Rojo. Enseñanza de la programación. [en línea] TE&ET, Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología, Universidad Nacional de Río Cuarto, Río Cuarto, Argentina, vol. 1, núm. 1, diciembre, 2006. Disponible en: http://teyet-revista.info.unlp.edu.ar/files/No1/09_Ensenanza_de_la_programacion.pdf

[12] D. Gayo, et al. Reflexiones y experiencias sobre la enseñanza de POO como único paradigma. [en línea] Universidad de Oviedo. Disponible en: http://www.di.uniovi.es/~dani/publications/jenui03.pdf

[13] A. Chaves, et al. Una Experiencia Exitosa en la Enseñanza de Fundamentos de Programación en Ingeniería de Sistemas. Reunión Nacional y Expoingeniería ACOFI 2009, Santa Marta, septiembre. 2009.

[14] R. Timarán, et al. Un Cambio de Paradigma en la Enseñanza de Fundamentos de Programación en Ingeniería de Sistemas. Revista Educación en Ingeniería, junio, 33 – 37, 2009.

[15] Y. Pérez y L. López, Multiparadigma en la Enseñanza de la Programación [en línea]. Universidad Nacional de Comahue, Buenos Aires. 2009. Disponible en: http://www.ing.unp.edu.ar/wicc2007/trabajos/TIAE/153.pdf

[16] E. Dijkstra and W. Feijen. A Method of Programming. Addison-Wesley, Boston, USA. 1988.

[17] J. Van. Teaching Programming at various levels of formality. IFIP. 1985.

[18] A. Blanco y L. Moreno. EDApplets: Una Herramienta Web para la Enseñanza de Estructuras de datos y Técnicas Algorítmicas [en línea]. 2003. Disponible en: http://bioinfo.uib.es/~ joemiro/aenui/procJenui/Jen2004/ponencias/ ponencia48. pdf

[19] R. Llamosa, I. Guarín, G. Moreno y S. Baldiris S. Sistema Hipermedia Adaptativo Para la enseñanza de los Conceptos Básicos de la Programación Orientada a Objetos [en línea]. 2003. Disponible en: http://lsm.dei.uc.pt/ribie/docfiles/ txt2003326195840A016.pdf

[20] J. C. López. Algoritmos y Programación en la Educación Escolar [en línea]. 2008. Disponible en: http://www.eduteka.org/modulos.php?catx=9

[21] R. Pérez. Una Herramienta y Técnica para la Enseñanza de la Programación [en línea]. 2008. Disponible en: http://campusv.uaem.mx/cicos/imagenes/memorias/6tocicos2008/Articulos/Cartel%206.pdf

[22] A. Orozco y S. Londoño. Nuevas Herramientas y Metodologías en la Enseñanza de la Programación Usando Alice y Robot Scribble. Reunión Nacional y Expoingeniería ACOFI Septiembre 2009, Santa Marta.

[23] R. Trejo, et al. La Enseñanza de la Programación Dentro de un Modelo de Integración Curricular: la Experiencia del Proyecto Principia" [en línea]. 2003. Disponible en: http://lsm. dei.uc.pt/ribie/docfiles/txt200312121727 24TCI15.pdf

[24] A. Chaves, et al. ICD-Chapin: Intérprete de comandos para diagramas N-S. Primer Congreso Internacional de Gestión Tecnológica e Innovación, Universidad Nacional de Colombia, Bogotá, 2008.

[25] J. Cerón y A. Nárváez. ICD-CHAPIN: Intérprete de Comandos para Diagramas Nassi - Shneiderman (N-S) Versión 2.0. (Trabajo de grado