Problemas de enseñanza y aprendizaje de los fundamentos de programación[*]

Problems of teaching and learning the basics of programming

Problemas de ensino e aprendizagem dos fundamentos de programação

Fecha de recepción: 3-02-2016
Fecha de aceptación: 10-04-2016

Disponible en línea: 30 de junio de 2016

DOI: http://dx.doi.org/10.18359/reds.1701

Jesús Insuasti[**]

Cómo citar este artículo:
Insuasti, J. (2016) Problemas de enseñanza y aprendizaje de los fundamentos de programación. *Revista educación y desarrollo social, 10* (2), 234-246. DOI: org/10/18359/reds.1701

[*] Artículo de revisión como parte de la reconstrucción del estado del arte para la investigación Computer Science Curricula based on the Semat Essence Kernel, en el marco de estudios del Doctorado en Ingeniería: Sistemas e Informática de la Universidad Nacional de Colombia, sede Medellín.

[**] Profesor Tiempo Completo del Departamento de Sistemas de la Universidad de Nariño (Pasto–Colombia), Ingeniero de Sistemas–Universidad de Nariño, Master of Science in Internet Systems–The University of Liverpool (Reino Unido). Grupo de investigación GALERAS.NET, correo electrónico: insuasty@udenar.edu.co

Resumen

Dentro de los escenarios relacionados con las ciencias computacionales, las actividades de aprendizaje asociadas a la programación de computadoras han sido reconocidas con alto grado de dificultad, según los antecedentes revelados en el presente artículo de revisión. Con esta situación, que al parecer es bastante común en el ámbito global, las causas que generan dicha problemática se relacionan con determinadas características que suceden dentro del aula de clase. Ciertas habilidades cognitivas son relevantes al momento del aprendizaje de los fundamentos de programación, tales como la capacidad de abstracción, una buena aptitud lógico-matemática y la facilidad para la resolución de problemas de orden algorítmico. En adición, factores de motivación son necesarios al momento de enfrentar las temáticas asociadas a los fundamentos de programación dentro de los escenarios de práctica. El contenido de esta revisión involucra experiencias en diferentes zonas del planeta, cuyo interés apunta a develar los orígenes del problema. Finalmente se elabora una reflexión en la búsqueda de posibles soluciones, y donde se abre el espacio de actuación a una nueva orientación basado en el núcleo de la Esencia de Semat.

*Palabras clave***:** Problemas; aprendizaje; programación; Semat.

Abstract

Among the scenarios related to computer science, learning activities associated with computer programming have been recognized with a high degree of difficulty, according to the information revealed in this review article. With this situation, which apparently is quite common at the global level, the causes of this problem are related to certain characteristics that happen inside the classroom. Certain cognitive skills are relevant at the time of learning the basics of programming, such as the ability of abstraction, a good logical-mathematical aptitude and ability to solve algorithmic order problems. In addition, motivators are necessary when faced with the issues associated with programming fundamentals in practice settings. The content of this review involves experiences in different areas of the planet, whose interest aims to uncover the origins of the problem. Finally, a reflection is made in the search for possible solutions, and where the performance space is opened to a new orientation based on the core of the Essence of Semat.

Keywords: Problems; learning; apprenticeship, programming, Semat.

Resumo

Dentro dos cenários relacionados com a informática, as atividades de aprendizagem ligadas à programação de computadores têm sido reconhecidas com um alto grau de dificuldade, segundo os antecedentes revelados no presente artigo de revisão. Com esta situação, que aparentemente, é bastante comum no âmbito global, as causas que geram dita problemática estão relacionadas com determinadas características que acontecem dentro da sala de aula. Certas habilidades cognitivas são relevantes no momento da aprendizagem dos fundamentos de programação, tais como a capacidade de abstração, uma boa aptidão lógico-matemática e a facilidade para a resolução de problemas de ordens algorítmicos. Além disso, fatores motivadores são necessários, no momento de se confrontar com os temas associados com os fundamentos de programação dentro dos cenários de prática. O conteúdo desta revisão envolve experiências em diferentes áreas do planeta, cujo interesse tem o objetivo de descobrir as origens do problema. Finalmente, é elaborada uma reflexão na busca de possíveis soluções, e onde se abre o espaço de atuação para uma nova orientação baseado no núcleo da Essência da Semat.

Palavras-chave: Problemas; Aprendizagem; Programação; Semat.

Introducción

La programación de computadoras es considerada a menudo una tarea difícil debido a la complejidad involucrada en ella. Existen estudiantes que no logran adquirir las habilidades necesarias para la programación, incluso después de la terminación de un curso de fundamentos de programación en las Ciencias Computacionales. Algunas investigaciones indican que las razones para no lograr los niveles de aprendizaje deseados, pueden ser debido a la complejidad de la sintaxis del lenguaje y los conceptos de programación; la carga cognitiva implicada en el aprendizaje de programación; el mal diseño de los objetos de saber, y la falta de habilidades cognitivas propias para la solución de problemas.

El presente trabajo realiza una exploración de la problemática en la enseñanza y en el aprendizaje de los cursos de fundamentos de programación, mediante una revisión documental en esta materia. Es importante anotar que la mayoría de las investigaciones han sido publicadas a través de ACM –*Association for Computing Machinery*–, considerada como una de las fuentes de mayor autoridad en el conocimiento relacionado a las Ciencias Computacionales a nivel mundial, incluyendo la educación de dichas ciencias. Posteriormente, se presenta una revisión de soluciones planteadas a esta problemática desde el campo de la didáctica específica de las Ciencias Computacionales. Con esto, se realiza un viaje a través de experiencias documentadas donde se develan los posibles orígenes de las dificultades de aprendizaje en materia de fundamentos de programación de computadoras.

Hallazgos acerca del problema

En el mundo de la academia de las Ciencias Computacionales, es común observar a la programación como un arte donde la creatividad y el ingenio son factores clave del éxito. Dicho arte contempla conocimientos en lenguajes de programación y herramientas asociadas, así como habilidades cognitivas orientadas a la solución de problemas. Desde esta óptica, un enfoque común en la enseñanza de los fundamentos de programación es abordar primero lo básico del lenguaje (o de los lenguajes) de programación, para luego guiar a los estudiantes a través de estrategias donde se contemple la totalidad del proceso de programación de computadoras.

La enseñanza de los fundamentos de programación, es un punto clave en la formación de profesionales en Ciencias Computacionales. Pero existe evidencia que sugiere cómo aprender a programar no es una tarea fácil; así lo expresan Soloway y Spohrer en su libro. Estos autores afirman que la creación y el control de ambientes y soluciones computacionales a través de la programación, son cosas que para un individuo pueden ser difíciles de realizar (1989, p. 2).

La prueba de esta afirmación puede ser constatada a través de las altas tasas de deserción y cancelación de cursos de

fundamentos de programación. Para citar un ejemplo, un estudio estima que entre un 25 a 80 por ciento de los estudiantes en Estados Unidos abandonan sus primeras clases de programación debido a la dificultad que enfrentan para aprender a programar (Carter & Jenkins, 2002). De igual forma, la enseñanza de un primer curso de programación ha sido objeto de numerosos estudios (Ali & Mensch, 2008; Kelleher & Pausch, 2005) y todos ellos presentan un común denominador en materia del aprendizaje, dada la forma de abordar los saberes. Con esto se confirma la noción de que aprender a programar se considera una tarea difícil para la mayoría de los estudiantes, y ha sido una de las principales razones de su deserción (Anewalt, 2008; Porter & Calder, 2004) y para los profesores de Ciencias Computacionales, esta cuestión de dificultad en los cursos de fundamentos de programación, ha sido objeto de debate.

Adicionalmente hay un amplio reconocimiento entre académicos en Ciencias Computacionales, sobre la necesidad de hacer algo para contrarrestar este problema; tal es el caso de los manifiestos de ACM en materia de reformas curriculares y planteamientos de didácticas (ACM, IEEE & AIS, 2005). Varios programas a nivel mundial han adoptado medidas enérgicas para compensar estos puntos de dificultad; algunos cambiaron el idioma en que se enseñan normalmente los cursos de fundamentos de programación; se realizaron cambios en el libro de texto guía a utilizar o se tomaron medidas adicionales en otros aspectos de la academia; sin embargo, el problema de la dificultad en el aprendizaje de los cursos de fundamentos de programación persistió (Marreno & Settle, 2005).

¿Por qué realmente es difícil programar? Quizás esta pregunta la han formulado (si no todos) la mayoría de los profesores que están (o estuvieron) a cargo de los cursos de fundamentos de programación. Sin duda la respuesta a dicho interrogante aún no ha sido encontrada en su totalidad. A continuación se presentan algunos factores que, dados los antecedentes investigativos, influyen negativamente en el aprendizaje de los fundamentos de programación en el ámbito de las Ciencias Computacionales a nivel global.

Un estudio logró identificar algunos factores que dificultan el aprendizaje de la programación en Ciencias Computacionales. Baldwin y Kulijis (2001) por ejemplo expresan: "*la mayoría de los estudiantes […], encuentran difícil y compleja la tarea cognoscitiva relacionada a la programación de computadoras" y explicaron: "el aprendizaje demanda complejas habilidades cognitivas tales como la planificación, razonamiento y resolución de problemas en programación de computadoras*" (p. 1). Con esto se entiende que existen habilidades de pensamiento que, de no haber sido desarrolladas previamente, se constituyen en factores negativos para el aprendizaje. Entre algunas habilidades de pensamiento se destacan: la capacidad de atracción, la facilidad de análisis (descomposición en partes), la destreza para la síntesis

(la re-composición de un todo); todas las anteriores son habilidades cognitivas fundamentales en la resolución de problemas.

Otros estudios proporcionan un análisis más exhaustivo de los factores que contribuyen a esta dificultad. Dann, Cooper y Pausch (2006) enumeran cuatro factores que contribuyen a la dificultad en el aprendizaje de los fundamentos de programación: mecanismos frágiles en la creación de programas de computadora, en particular el uso de la sintaxis de los lenguajes de programación; la incapacidad para ver el resultado de los cálculos a la par cuando un programa de computadora se ejecuta; la falta de motivación para la programación, la dificultad de comprensión de la lógica compuesta, y el desconocimiento en las técnicas de diseño. Lo interesante de este estudio en particular, es la utilización de un lenguaje nuevo de programación, donde el factor de motivación se encuentra en cada escenario de uso del lenguaje. El lenguaje en particular es Alice y fue creado para propósitos educativos en forma exclusiva; dicho lenguaje proporciona los mecanismos para facilitar el aprendizaje de la programación en ambientes tridimensionales basados en guiones y narraciones de historias cortas (Carnegie Mellon University, 2013).

En la fase de planeación para la creación de un programa de computadora, el objetivo de la planeación es desarrollar un buen grado de comprensión de los conceptos de los fundamentos de programación como la estructura y el flujo total del programa. En la jerga de los fundamentos de programación de computadoras, el término 'estructura' se usa frecuentemente y se practica al momento de escribir programas (que corresponde a la fase de codificación). En realidad, la estructura otorga a un programa de computadoras un orden de cómo debe ejecutarse en una computadora. En la práctica, la enseñanza del concepto de estructura no ha sido tan sencilla para los estudiantes que no estén familiarizados con la programación de estas. Adicionalmente, la evolución misma de los lenguajes de programación ha forzado a generar sólidas estructuras a fin de entender cómo fueron codificados los programas de computadoras, y esto constituye un reto adicional en la enseñanza de los fundamentos de programación (Schneider, 1999). He aquí un factor que dificulta este aprendizaje dado que a muchos estudiantes les resulta difícil entender cómo un programa de computadora realiza comparaciones, saltos al interior del código, ciclos e iteraciones, y los objetos pueden realizar herencia y polimorfismo. Entonces, la forma de abordar estas temáticas dentro de las aulas de clase por parte de los profesores influye en forma directa en los aprendizajes de los estudiantes, dada la complejidad de los objetos de saber.

Frente a esta situación se tiene entendido que los estudiantes de pregrado que se matriculan o se inscriben en los cursos de fundamentos de programación en Ciencias Computacionales no tienen experiencia previa en programación de computadoras; así pues, la experiencia en

programación no es un requisito previo. Aquí es posible identificar una brecha entre aquellos estudiantes que a nivel de bachillerato (colegio o educación básica secundaria) han estudiado algunas materias referentes a la programación de computadores, y aquellos estudiantes que no tuvieron esa oportunidad en su ciclo de formación previa. La falta de experiencia previa en programación de computadoras no parece ser un problema; sin embargo, sí es un problema el bajo desarrollo de habilidades para la resolución de problemas.

Dunican (2002) revela que los objetos de saber ofrecidos en las escuelas secundarias no incluyen los módulos de lógica o solución de problemas, poniendo a los estudiantes en una situación difícil cuando se matriculan en cursos de fundamentos de programación a nivel universitario, específicamente en las carreras asociadas a las Ciencias Computacionales. De igual forma, Stamouli y Doyle & Huggard (2004), también señalaron la falta de continuidad en los estudios de aquellos sujetos que salen de las escuelas secundarias e ingresan al primer año de estudios universitarios; en este apartado se evidencia que esa discontinuidad interfiere significativamente en los aprendizajes de los estudios universitarios.

Aunque el nivel de 'alfabetismo' en computación y TIC puede ser alto entre algunos de los estudiantes que ingresan a carreras asociadas a las Ciencias Computacionales (dado que previamente han recibido formación al respecto en la básica secundaria), la mayoría de ellos tiende a carecer de experiencia específica en programación de computadoras. Esto incluye no solo las fases de diseño y construcción, sino también las tareas rutinarias como compilar, depurar o ejecutar un programa de computadora, o, de hecho, una comprensión básica de un modelo computacional con sus componentes de hardware y software. Esta falta de comprensión de un modelo mental de una computadora (es decir, cómo el ser humano ve a una computadora desde el punto de vista meramente conceptual) a menudo resulta frustrante cuando los resultados no muestran lo que el estudiante había planeado previamente (Ben-Ari, 1998).

Complementando lo anterior, otra dificultad que enfrentan los estudiantes de programación, es la necesidad de imaginar y comprender términos abstractos que no tienen equivalentes en la vida real: ¿Cómo se relaciona una variable, un tipo de datos o una dirección de memoria a un objeto de la vida real? De esta manera, varios de los conceptos de programación de computadoras tienden a ser difíciles de entender (Dunican, 2002). En consecuencia, algunos estudiantes de los cursos de fundamentos de programación asumen una actitud de 'odio' en su esfuerzo por comprender los conceptos de la programación de computadoras (Stamouli et al., 2004; Thomas, Ratcliffe, Woodbury & Jarman, 2002).

Nuevamente la forma en que los objetos del saber son abordados, impacta en el

aprendizaje de los estudiantes, dado que conceptualmente no se ha logrado 'empatar' con elementos de la vida real; punto clave para los estudios en didáctica, donde se aborde esta problemática de índole conceptual.

Buscando soluciones: estudios en didáctica para los cursos de fundamentos de programación en Ciencias Computacionales

Si bien es cierto que se han logrado identificar algunos factores que inciden negativamente en el aprendizaje de los objetos de saber en los cursos de fundamentos de programación para las Ciencias Computacionales, la producción a manera de propuestas didácticas para abordar dicha problemática no ha sido prolífera. Presentamos algunas que se han formulado, y las medidas sugeridas para simplificar este proceso de aprendizaje. Se pueden clasificar en sugerencias simples (tales como cambiar el lenguaje de programación) y sugerencias más detalladas, que tienen que ver con el modelo conceptual y el paradigma de la enseñanza de los lenguajes de programación.

Herbert (2007) señaló que para hacer más fácil el aprendizaje de los fundamentos de programación, se deben mantener tres elementos: minimizar la sintaxis de programación, proporcionar retroalimentación visual sobre la ejecución de los programas de computadoras, acortar el ciclo creativo de conceptualización, además de mejorar la implementación y los resultados obtenidos tras la ejecución de un programa de computadora. En un estudio que explica sobre el modelo conceptual y la comprensión del estudiante sobre un lenguaje de programación, Baldwin y Kulijas (2001) señalaron lo siguiente:

> "*Se ha argumentado que los modelos conceptuales pueden servir para mejorar la comprensión conceptual de los estudiantes de programación. Los métodos utilizados para mejorar el desarrollo de modelos mentales precisos incluyen: diseño de la interfaz para que los usuarios pueden interactuar activamente con ella; uso de metáforas y analogías para explicar los conceptos; y el uso de relaciones espaciales para que los usuarios puedan desarrollar capacidades para la simulación mental*"(p.1).

Dann, Cooper y Pausch (2006), resaltaron tres temas que los estudiantes deben aprender en los cursos de fundamentos de programación: pensamiento algorítmico y expresión, abstracción y apreciación de la realidad en detalle. Adams (2008), explicó que para resolver los problemas relacionados con cursos introductorios de programación, el profesorado debe incluir ejemplos de la vida real en los cuales están participando, a fin de capturar la atención del estudiante de hoy.

En cuanto a la forma como son abordados los objetos del saber dentro del aula, Herbert (2007) aclaró que la mejor manera de enseñar las ideas de los fundamentos de programación, es mediante

una exposición 'suave' ante los estudiantes, para luego añadir más detalles hasta abarcar en profundidad un objeto de saber. Este tipo de enfoque para aprender los fundamentos de programación se conoce como el 'enfoque de espiral'. El proceso puede ser largo y a veces tedioso; por lo tanto los profesores necesitarán motivar a los estudiantes durante todo el camino.

Desde el punto de vista tecnológico, los nuevos lenguajes de programación han evolucionado a tal punto que sus interfaces integradas de desarrollo (del inglés: IDE – *Integrated Development Environment*) son una ayuda significativa en el procesos de construcción de programas de computadora. Estas herramientas ayudan a mejorar el uso de la sintaxis de los lenguajes de programación y potencian la reutilización de activos de software. Con esto se tiene que las diferentes propuestas de orden didáctico, siempre sugieren el uso de este tipo de herramientas para facilitar los procesos de aprendizaje en la construcción de programas de computadora.

Por otro lado, Guibert, Girard y Guitet (2004) hicieron hincapié en la experiencia positiva del uso de programación (del inglés *PbE – Programming by Example*), donde los programadores diseñan métodos para proporcionar retroalimentación continua durante la ejecución del programa. El hecho de proveer dicha retroalimentación involucra al estudiante en él, haciéndolo a la vez consciente de lo que se está produciendo durante la ejecución del mismo en la computadora. En este acápite, es pertinente la utilización de depuradores (*debuggers*) integrados al entorno de programación, a fin de hacer conciencia sobre lo que el programa hace paso a paso.

En un estudio realizado por Hartman, Nievergelt Reichert (2001), fue sugerido el uso de 'Máquinas de Estados Finitos' para la enseñanza de los fundamentos de programación, y se señaló además que "*se debe ver a la programación practicada como un ejercicio educativo, libre de la preocupación utilitaria es mejor aprendida en un ambiente de juego, diseñada para ilustrar los conceptos seleccionados en la configuración más simple*" (p. 1). El estudio sugirió introducir los fundamentos de programación para principiantes en un entorno de juego, donde a través de acciones limitadas es posible aprender a controlar rutinas simples tales como condicionales, ciclos e iteraciones. El propósito del uso de las máquinas de estado finito y los juegos de azar, es estimular el aprendizaje de conceptos a través la lúdica y la motivación.

Así mismo, la articulación del mundo real con los fundamentos de programación, es un elemento crucial para mejorar el aprendizaje de sus objetos de saber. El uso de cosas que se asemejan a instancias de vida, puede ayudar a la comprensión conceptual de las características mencionadas en dichos cursos de fundamentos de programación. En varios casos, los lenguajes de programación introducen objetos que están cerca de los reales y pueden ser utilizados para facilitar el aprendizaje de

los nuevos conceptos de programación orientada a objetos (OOP – *Object-Oriented Programming*). Por ejemplo, una persona puede ser considerada como un objeto; la persona tiene propiedades (ancho, altura), se pueden crear métodos (correr, caminar) y funciones para dichos objetos. De esta forma, un programa que muestra las manipulaciones de diversos objetos basados en la persona (ancho, alto), puede ser más comprensible que aquellos programas estáticos que expresan manipulación de texto y cálculos sencillos.

Saliendo momentáneamente del campo de la didáctica para entrar al mundo curricular, se han creado a través de los años diversos tipos de intervenciones para ayudar a los estudiantes a desarrollar habilidades de programación. Las intervenciones variaron entre los cambios y el currículo, la pedagogía y la evaluación, para conseguir un apoyo adicional a los nuevos estudiantes de los cursos de fundamentos de programación. En este sentido, Van Roy, Armstrong y Flatt y Magnusson (2003), exitosamente basaron las unidades de programación en conceptos en lugar de tipos de lenguajes o paradigmas individuales (tales como programación orientada a objetos, programación lógica o la programación funcional). El hecho de haber enseñado durante dos años con este enfoque en cuatro universidades, ha permitido descubrir que los estudiantes "*razonen de manera amplia y profunda sobre el diseño de sus programas, su corrección y su complejidad*" (p. 270).

Dos enfoques contradictorios para el diseño curricular han sido utilizados y probados en diversas instituciones: el enfoque de enseñanza de objetos en primera instancia, y el enfoque de la enseñanza de la programación estructurada; estos dos enfoques se han divulgado como casos exitosos. Sin embargo, Sheard y Hagan (1998) observaron que los estudiantes "*comenzaron a sentirse perdidos... casi al mismo tiempo cuando se introdujo un nuevo paradigma*" a la unidad (p. 315). En consecuencia se decidió utilizar primero el enfoque bottom-up (programación estructurada primero) más tradicional, e introducir luego los conceptos orientados a objetos después de que los estudiantes han ganado una comprensión en materia de expresiones, declaraciones, parámetros, etc. Este fue uno de los cambios introducidos a la unidad desde el punto de vista curricular y no didáctico, y como resultado: "*fue encontrado un aumento significativo en el rendimiento después del cambio curricular en materia de los cursos de fundamentos de programación*" (Sheard & Hagan, 1998, p. 319).

Volviendo al campo de la didáctica, una de las técnicas de enseñanza empleada para el aprendizaje de los fundamentos de programación se basa en la analogía. Esta técnica es particularmente útil al instruir en los fundamentos de programación, tales como entrada/salida, tipos de datos, búsquedas, clasificación, etc; la analogía utiliza ejemplos ilustrativos de conceptos que los estudiantes han visto antes, de tal suerte que dichos conceptos

familiares se relacionan con los nuevos conceptos. En una analogía, el concepto familiar es identificado como la fuente y el nuevo concepto como el objetivo, y cuando se hace analogía, la fuente se asigna al objetivo (Blanchette & Dunbar, 2000). Dunican (2002) describe varias analogías, por ejemplo: el uso de juguetes infantiles para enseñar las declaraciones de misión; la utilización de cajas para determinar el número más pequeño y más grande en una lista, y el manejo de un casillero de correspondencia para explicar el concepto de manipulación de datos en una matriz.

Otra importante faceta didáctica es el concepto de pertinencia: los estudiantes deben ver un fin en lo que están aprendiendo. Hagan y Sheard (1998) informan sobre la respuesta positiva de estos después de los ejercicios de programación de computadoras con atractivas interfaces gráficas al estilo de los videojuegos y cuando fueron utilizadas dichas interfaces para mostrar los beneficios del paradigma orientado a objetos. Esta ilustración proporcionó una oportunidad para explicar las ventajas de diseño y programación orientada a objetos sobre otros estilos de programación de aplicaciones complejas. Una vez más, los objetos de saber son transformados en objetos atractivos y motivadores de aprendizaje.

Para finalizar, otro enfoque se basa en el uso de tecnología para la enseñanza. Clancy, Titteron, Ryan, Slotta, y Linn (2003), describen sus esfuerzos por desarrollar un modelo basado en laboratorio para la enseñanza de las Ciencias Computacionales, el cual incluye tres componentes: un constructor de curso en línea para el profesor, un entorno de aprendizaje basado en Web para la entrega de todas las actividades del estudiante, y un portal de curso que sirvió como un sistema de gestión de aprendizajes. La evaluación del sistema ha demostrado que se ha mejorado el rendimiento de los estudiantes en el desempeño de fundamentos de programación y que lo encontraron agradable. Sin embargo el nuevo modelo no tenía ningún impacto en la tasa de deserción.

Conclusiones

Si bien es cierto que la evolución de los lenguajes de programación ha sido vertiginosa en las últimas décadas, los avances en materia de estrategias de enseñanza y de aprendizaje han experimentado cierto retraso. Existen múltiples manifestaciones de escenarios para la enseñanza y el aprendizaje de los fundamentos en programación de computadoras, así como herramientas tecnológicas que de alguna forma complementan los quehaceres dentro del aula de clase; sin embargo, hay una marcada tendencia hacia el uso del enfoque instruccional, donde prima la enseñanza a través del ejemplo y el desarrollo de ejercicios subsecuentes.

Al parecer, la utilización de otros enfoques didácticos que involucren escenarios externos al aula de clase magistral y a los laboratorios de computadoras, se

ha contemplado en forma incipiente. Lo que es importante recalcar en este estudio es el desarrollo de esfuerzos aislados los unos de los otros, a fin de gestar un 'repositorio de experiencias exitosas' que debería conformarse a través de un estudio sistemático y global, frente a la problemática planteada.

Con relación al concepto de repositorio de experiencias exitosas, varias iniciativas han sido desarrolladas a manera de plataformas LMS (del inglés *Learning Management System*). No obstante, la creación de contenidos digitales como parte fundamental de los objetos virtuales de aprendizaje que representan un valioso aporte al ejercicio de la enseñanza, suele quedarse en la simple publicación de recursos, que sin una adecuada apropiación didáctica se podría desperdiciar el potencial de los materiales creados. Por tal razón, además de la base de datos de recursos digitales, es menester utilizar un repositorio de experiencias exitosas, donde se narre la vida dentro del aula y de forma explícita se pueda evidenciar cómo el profesor realiza el ejercicio de docencia, más allá de la publicación de recursos digitales que puedan ser utilizados. Así podría enriquecerse el conocimiento sobre la enseñanza de los fundamentos de programación, desde una mirada de las experiencias exitosas.

He aquí una oportunidad donde las ciencias de la educación y las computacionales, podrían explorar sus potencialidades en pro de mejorar las condiciones de la enseñanza y del aprendizaje de los fundamentos de programación de computadoras.

Referencias

ACM, IEEE & AIS (2005). Computing Curricula: The Overview Report 2005. En Internet: http://www.acm.org/education/education/curric_vols/CC2005-March06Final.pdf, Retrieved September 28, 2013.

Ali, A., & Mensch, S. (2008). Issues and challenges for selecting a programming language in a technology update course. Proceedings of the Information Systems Education Conference, Phoenix, AZ 2008. Retrieved September 28, 2013 from http://isedj.org/isecon/2008/020/index.html.

Anewalt, K.(2008). Making CS0 fun: An active learning approach using toys, games and Alice. Journal of Computing Sciences in Colleges, 23(3), 98-105. September 28, 2013 from ACM Digital Library http://www.acm.org/dl.

Ben-Ari, M. (1998). Constructivism in computer science education. Proceedings of the 29th SIGCSE Technical. Symposium on Computer Science Education, 257-261.

Baldwin, L.P., & Kuljis, J. (2001). Learning programming using program visualization techniques. Proceedings of the 34th Hawaii International Conference on System Sciences – 2001. Retrieved September 29, 2013 from IEEE Computer Society Digital Library http://www.computer.org/portal/.

Blanchette, I., & Dunbar, K. (2000). How analogies are generated: The roles of

structural and superficialsimilarity. Memory and Cognition, 28, 108-124.

Carnegie Mellon University (2013). Alice's Home Page, An Educational Software that teaches students computer programming in a 3D environment. Retrieved September 28, from http://www.alice.org/index.php?page=what_is_alice/what_is_alice.

Carter, J., & Jenkins, T. (2002). Gender differences in programming?. Proceedings of the 7th Annual Conference on Innovation and Technology in Computer Science Education. September 28, 2013 from ACM Digital Library http://www.acm.org/dl.

Clancy, M., Titteron, N., Ryan, C., Slotta, J., & Linn, M. (2003). New roles for students, instructors, andcomputers in a lab-based introductory programming course. Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, 132-136.

Dann, W., Copper, S., & Pausch, R. (2006). Learning to program with Alice. Upper Saddle River, NJ: Prentice Hall.

Dunican, E. (2002). Making the analogy: Alternative delivery techniques for first year programming courses. In J. Kuljis, L. Baldwin & R. Scoble (Eds.), Proceedings from the 14th Workshop of the Psychology of Programming Interest Group, Brunel University, June 2002, 89-99.

Guibert, N., Girard, P., & Guittet, L. (2004). Example-based programming: A pertinent visual approach for learning to program. Proceedings of the Working Conference on Advanced Visual Interfaces, 358 – 361. Retrieved September 29, 2013 from ACM Digital Library http://www.acm.org/dl.

Hartman, W., Nievergelt, J., & Reichert, R. (2001). Kara, finite state machines, and the case for programming as part of general education. Proceedings of the IEEE Symposium on Human-Centric Computing Languages and Environment (HCC01) Retrieved September 29, 2013 from IEEE Computer Society Digital Library http://www.computer.org/portal/.

Herbert, C. (2007). An introduction to programming with Alice. Boston, Massachusetts: Course Technology.

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environment and languages for novice programmers. ACM Computing Surveys, 37(2), 83-137. Retrieved September 28, 2013 from ACM Digital Library http://www.acm.org/dl.

Marrero, W., & Settle, A. (2005). Testing first: Emphasizing testing in early programming courses. Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education, 4-8. Retrieved September 28, 2013 from ACM Digital Library http://www.acm.org/dl

Porter, R., & Calder, P. (2004). Patterns in learning to program: An experiment?. Proceedings of the Sixth Conference on Australasian Computing Education, 30, 241-246. Retrieved September 28, 2013 from ACM Digital Library http://www.acm.org/dl

Schneider, D. (1999). An introduction to programming using Visual Basic 6.0. Upper Saddle, River, NJ: Prentice Hall.

Sheard, J., & Hagan, D. (1998). Experiences with teaching object-oriented concepts to introductory programming students using C++. Technology of Object-Oriented Languages and Systems-TOOLS 24, IEEE Technology, 310-319.

Soloway, E. & Spohrer, J. (1989). Studying the Novice Programmer, Lawrence Erlbaum Associates, Hillsdale, New Jersey. 497 p.

Stamouli, I., Doyle, E., & Huggard, M. (2004). Establishing structured support for programming students. Proceedings of the 34th ASEE/IEEE Frontiers in Education Conference, Savannah, GA, October 2004.

Thomas, L., Ratcliffe, M., Woodbury, J., & Jarman, E. (2002). Learning styles and performance in the introductory programming sequence. Proceedings of 33rd SIGCSE Technical Symposium, 34, 33-37.

Van Roy, P., Armstrong, J., Flatt, M., & Magnusson, B. (2003). The role of language paradigms in teaching programming. Proceedings of the 34th SIGCSE Technical Symposium on Computer science Education, 269-270.