Enseñanza de la Programación: un tema en la agenda académica para repensar año a año

Claudia Carina Fracchia, Ana Alonso de Armiño, Adair Martins

Facultad de Informática, Universidad Nacional del Comahue,
Buenos Aires 1400, Neuquén
{carina.fracchia, ana.alonso, adair.martins} @fi.uncoma.edu.ar

Resumen. Aprender a programar implica un proceso mental complejo, que requiere que el alumno pueda comprender el problema a resolver y entender el procedimiento mediante el cual puede llegar a una solución. Las TIC pueden ayudar a que los estudiantes mejoren sus aprendizajes, pero para lograrlo es necesario configurar un nuevo escenario donde se fortalezcan las relaciones entre los profesores, los alumnos y los contenidos a enseñar. En este trabajo se analizan diferentes recursos TIC orientados a la enseñanza de la programación, sus ventajas y requisitos a la hora de incorporarlos en la práctica educativa.

Palabras clave: Enseñanza de Programación, TIC, Aprendizaje Colaborativo

1 Introducción

Este trabajo se enmarca en la investigación realizada dentro de los proyectos de investigación "Simulación y Métodos Computacionales En Ciencias y Educación", en la línea "Desarrollo y Uso de TIC" y "Aspectos Avanzados de la Programación de Computadoras", ambos pertenecientes a la Facultad de Informática de la Universidad Nacional del Comahue (UNCo).

La programación de computadoras se utiliza en niños de edades tempranas para fomentar el desarrollo de habilidades de orden superior, tales como la destreza para resolver problemas. Mientras que en la educación básica se tiende al empleo de ambientes de programación fáciles de usar, muchos de ellos basados en el lenguaje Logo[1], por lo general en las carreras universitarias de computación se emplean en los primeros años entornos que demandan el uso de enfoques de programación tales como el estructurado, funcional y orientado a objetos. Generalmente, como es el caso de nuestra Institución, se prioriza el uso de software libre y entornos de programación amigables.

Desde el punto de vista educativo la programación de computadoras posibilita activar una amplia variedad de estilos de aprendizaje, desarrollar el pensamiento algorítmico, y en muchos casos la creatividad e innovación. Posibilita a los estudiantes encontrar diversas maneras de abordar problemas y plantear soluciones, al

[1] Logo, http://el.media.mit.edu/logo-foundation/pubs/logoupdate/index.html

tiempo que desarrollan habilidades para: visualizar caminos de razonamiento divergentes, anticipar errores, y evaluar rápidamente diferentes escenarios mentales. Los problemas del tipo pensamiento lateral permiten que se exploren y consideren la mayor cantidad de alternativas para solucionar un problema, favoreciendo además que el estudiante explore, escuche y acepte, en algunos casos, diferentes puntos de vista.

En las situaciones donde la resolución de un problema involucre búsqueda de información, será necesario e indispensable ayudar a los estudiantes a desarrollar la Competencia para Manejar Información. El estudiante debe poner en práctica habilidades, conocimientos y actitudes, para identificar lo que necesita saber sobre un tema específico en un momento dado, realizar búsquedas efectivas de la información requerida, además de poder determinar si es pertinente al mismo [1].

Aprender a programar implica un proceso mental complejo, que requiere que el alumno pueda comprender con claridad el problema a resolver o simular por medio de una computadora, y entender el procedimiento mediante el cual se llegará a la solución deseada. No es fácil ni su enseñanza ni su aprendizaje, requiere dedicación y esfuerzo por parte de alumnos y profesores, en particular durante los primeros años para así asentar buenos hábitos de programación. Una dificultad que se observa es que los alumnos tratan en primer lugar de aplicar directamente al problema conocimientos de experiencias pasadas, luego, después de intentos frustrados se ven obligados a modificar su solución para ajustarse a los requisitos del problema. Comúnmente se comienzan a realizar cálculos o construcciones sin haber comprendido el problema. Tales errores se cometen con frecuencia, y a veces suelen deberse a la falta de comprensión o de interés por parte del alumno. Una estrategia ante estas situaciones es cambiar el vocabulario, plantear la misma pregunta en diferentes formas, dado que muchas veces el propósito de estas preguntas es concentrar la atención del alumno sobre la incógnita, datos relevantes o relaciones entre datos.

Estudios recientes[2] han mostrado que cuando se toman apuntes en papel en contraste con el uso de un procesador, se logra que el alumno retenga más información y logre una mayor comprensión de los temas tratados. De la misma manera creemos que el diseño de algoritmos en papel versus el uso de una herramienta automatizada para diseño de algoritmos, permitirá una mayor comprensión por parte de los estudiantes.

Se han comparado los programas de estudio de materias introductorias a la programación de diferentes carreras ofrecidas en la UNCo y se ha observado que aunque el perfil del egresado es diferente, los contenidos trabajados en los respectivos planes de estudios y objetivos coinciden en gran parte. Entrevistas realizadas a docentes que dictan contenidos de programación en los primeros años universitarios en la UNCo, como así también docentes de Educación media que enseñan programación en 4to y 5to año, además de un análisis del material didáctico utilizado, ha permitido vislumbrar una similitud en la manera de abordar la enseñanza de programación. Primero se introduce a los alumnos en la resolución de problemas matemáticos, luego en base a estos se empiezan a definir los primeros algoritmos que serán implementados posteriormente en un lenguaje de programación. En una primera etapa se comienza trabajando con problemas simples que involucran cambios de

[2] http://www.psychologicalscience.org/index.php/news/releases/take-notes-by-hand-for-better-long-term-comprehension.html

representación de la información y la aplicación de diferentes estrategias de resolución. Se hace hincapié en que los estudiantes reflexionen sobre las actividades que realizan para solucionarlo (metacognición) y las agrupen de acuerdo a las etapas que contenga la estrategia de solución empleada.

Experiencias realizadas aplicando dinámica grupal, han permitido observar beneficios al retomar los errores cometidos por los estudiantes, o problemas de interpretación de enunciados, como una opción valiosa de retroalimentación. Mediante propuestas de generación de lluvias de ideas se perseguía que los estudiantes discutieran y compartieran posibles representaciones y estrategias de resolución, posibilitando la formulación de soluciones alternativas al problema dado, y de criterios para la selección de la que se consideraba óptima. Las actividades ofrecidas eran diagramadas para brindar a los estudiantes la posibilidad de discutir, hacer conjeturas, sacar conclusiones, defender ideas y escribir conceptualizaciones. Además de proporcionar oportunidades para realizar trabajo reflexivo y colaborativo [2].

2 ¿Cómo Motivar a los Nativos Digitales?

Nuestra meta como docentes debe ser el buen aprendizaje de nuestros alumnos, lo que requiere por parte de ellos la puesta en marcha de procesos que les permitan comprender, analizar, organizar, relacionar, reestructurar, sintetizar y aplicar la información que reciben. Para aprender algo sólidamente el alumno debe primero esforzarse por comprenderlo o entenderlo. Si logra expresarlo de forma organizada por escrito o verbalmente, tendrá la posibilidad de evaluar su propio aprendizaje con la certeza de que éste será duradero. Por medio del aprendizaje nuestros estudiantes adquieren y practican nuevas destrezas, actitudes y valores que les serán necesarios para vivir en un mundo que está en constante cambio.

El proceso educativo se desarrolla en un escenario que determinará los métodos y los estímulos con los que se lleva a cabo el aprendizaje, con respecto a esto, la tecnología ha permitido proveer diversos escenarios que facilitan los aprendizajes colaborativos, cooperativos, basados en proyectos y en metas. Si los alumnos aprenden cooperando podrán practicar regularmente competencias sociales y morales importantes, como por ejemplo el hábito de considerar los puntos de vista ajenos y la capacidad para el trabajo en equipo, a la vez que se aprenden los contenidos académicos fijados [3].

Las competiciones son herramientas poderosas para involucrar más a los estudiantes. En nuestra Institución se ha iniciado en el corriente año un proyecto de extensión en conjunto con instituciones de Enseñanza Media que trabajan programación en los dos últimos años, en el cual se realizaron Competencias de Programación hasta el momento implementadas en forma presencial. El mecanismo utilizado fue presentar un problema a grupos de alumnos, integrados por jóvenes de distintas escuelas, para ser resuelto. Posteriormente las soluciones eran evaluadas por un jurado. Se les asignaba un tiempo para la resolución y puntajes que consideraban la correctitud del algoritmo y el tiempo empleado. Está previsto en una etapa posterior realizar estas competencias mediante el ingreso a una plataforma que será la

encargada de habilitar los enunciados y realizar la corrección en forma automática. Hay que considerar que en estos casos no se evalúan buenas prácticas de programación, por ejemplo la elección de estructuras de control adecuadas, nombres de variables significativos, uso de comentarios, estructura del programa, modularización, etc. Es decir se realizan pruebas del tipo caja negra. En [4] se describe una investigación donde se muestra otro tipo de competencia en la que se trabaja con robots y animales, logrado como resultado fomentar la actitud colaborativa en vez de la competitiva y el aprendizaje basado en la motivación del alumno. Se logró aplicar el conocimiento a la resolución de tareas del mundo real mediante robots, constituyendo una herramienta de motivación muy efectiva y poderosa para mejorar el aprendizaje.

3 Metodologías y Software para el Diseño de Algoritmos

Por lo general los problemas que se abordan en la vida real no poseen una especificación simple y precisa, por esta razón se hace imprescindible introducir metodologías que faciliten el diseño de un modelo formal.

En las materias introductorias a la programación se utiliza el aprendizaje visual como método principal para enseñar habilidades del pensamiento, dado que las técnicas que utilizan formas gráficas para representar ideas e información ayudan a los estudiantes a clarificar y organizar su pensamiento. “Los diagramas visuales revelan patrones, interrelaciones e interdependencias además de estimular el pensamiento creativo” [5].

Estas técnicas ayudan a los estudiantes a organizar su pensamiento de forma lógica, permiten agilizar la codificación (o traducción) e identificar los errores más fácilmente. Entre sus ventajas se destacan que están libres de la sintaxis que agrega el lenguaje de programación, no contienen declaración de tipos de datos y se pueden utilizar como complemento de la documentación.

Resulta indispensable hallar una representación adecuada que permita describir un problema o fenómeno dado, es decir un lenguaje que a través de símbolos y combinaciones adecuadas, posibiliten representar algún aspecto de la realidad, o lo que tenemos en mente. “Buena parte de nuestros procesos mentales y psicológicos se reducen a descripciones constantes que hacemos respecto de la realidad que nos rodea, para lo cual requerimos asociaciones entre conceptos y elementos del lenguaje (símbolos) en su sentido más amplio” [6].

El uso de técnicas de diagramación resulta esencial para la comunicación de las personas involucradas en un proyecto y para clarificar las ideas de cada una de ellas. Facilitan describir procedimientos y acciones complejas que resultarían muy difíciles si sólo se usara texto. Muchas veces mediante el uso de una imagen se puede mostrar de forma concisa, precisa y clara las acciones a realizar, que si se usaran solamente palabras. La descripción realizada debe ser lo suficientemente completa para asegurar que un tercero pueda reproducirla, partiendo del mismo lugar y logrando así un conocimiento transmisible.

Con respecto a la documentación del programa, estas técnicas son utilizadas para definir y representar especificaciones del programa y de su diseño; por ejemplo cómo

se estructuran los datos y cómo es el control de flujo de la información. Por ejemplo algunos diagramas posibilitan mostrar la estructura de un programa, otros permiten observar los detalles internos tales como dónde se inicializa y referencia cada variable.

Para que todo lo descrito sea posible, los diagramas deben utilizar una notación consistente, utilizando elementos cuya interpretación resulte obvia (y familiar), y evitando introducir símbolos o términos nemotécnicos que no hayan sido explicados en la técnica de diagramación empleada.

Existen diversas técnicas de diagramación para programación estructurada, algunas son gráficas y utilizan diferentes símbolos para las distintas operaciones, otras se basan en texto y por último existen las llamadas híbridas. En el cuadro 1 se muestran las técnicas más utilizadas y las herramientas tecnológicas que pueden usarse como soporte.

		Características	Herramientas tecnológicas
Diagrama	*Warnier -Orr*	Uso de llaves para mostrar la descomposición jerárquica de los datos y actividades. Fáciles de aprender. Representan de manera gráfica la estructura jerárquica de un sistema, programa o estructura de datos. Los diagramas pueden leerse de izquierda a derecha, de arriba hacia abajo.	Wotree mindapp
	Diag. de flujo	Permiten al alumno representar y visualizar el circuito de información, desde su entrada como dato hasta el resultado dado en su salida, esclareciendo la secuencia de operaciones. Algunas aplicaciones permiten la edición, interpretación, el trabajo online u offline, individual y colaborativo, exportar a diferentes formatos, etc.	**Edición:** RFFlow, Flowchart 5.05., Gliffy **Edición e interpretación:** DFD 1.1, RAPTOR **Visualización:** PseInt, ECDIA
	Pseudocódigo	Diseño de un programa utilizando una forma de descripción narrativa. Fácil de leer y entender. Uso de palabras y frases del lenguaje natural, sujetos a determinadas reglas. Se pueden especificar instrucciones de entrada, de salida, de proceso, secuencias de control de flujo, acciones compuestas y comentarios. El uso de identación ayuda a visualizar la estructura lógica y jerárquica del programa. Se utiliza en la mayoría de los libros de texto de programación.	**Edición e interpretación:** Pseint SLE2 LPP ECDIA

Cuadro 1. Técnicas y herramientas para el diseño de algoritmos.

Las herramientas referenciadas se diferencian en el modo de trabajo (online-offline), tipo de actividades (individual-grupal), libres o software pago, y aspectos relacionados a la interfaz usuario (amigable, personalización, etc.), entre otros. Facilitan el diseño de algoritmos dado que por lo general trabajan con objetos visuales que pueden arrastrarse, reacomodándose en forma inmediata y automática.

Si bien pueden realizarse y usar adaptaciones de estas metodologías, es indispensable contar con especificaciones de antemano, lo que ayudará al alumno a ser consistente en sus realizaciones. Además le permitirá independizarse del lenguaje de programación a utilizar así como de los detalles de especificaciones de tipos de datos, y facilitando la comunicación y el trabajo en equipo. Algunas de las herramientas que soportan el trabajo colaborativo cuentan con mecanismos que permiten visualizar los aportes realizados por cada integrante de manera individual, lo que resulta fundamental para la evaluación que efectúa docente.

Una de las herramientas mencionadas es ECDIA (Entorno Colaborativo para el Diseño e Implementación de Algoritmos) [7]. Esta herramienta computacional desarrollada en el marco de una tesis de grado de la Licenciatura en Ciencias de la Computación, está basada en software libre, y su principal ventaja es que integra el trabajo con metodologías para el diseño de algoritmos y lenguajes de programación tales como C++ y Java. Sus funcionalidades básicas incluyen la edición colaborativa de algoritmos y su posterior codificación en un lenguaje seleccionado, interpretación y verificación tanto de algoritmos como del código generado, permitiendo la ejecución paso a paso y la confección de la traza resultante. Además, cuenta con mecanismos de comunicación para el envío y recepción de los diseños y programas realizados.

4 Uso de TIC en la Enseñanza de Programación

Las TIC permiten configurar nuevos escenarios para las relaciones entre los profesores, los alumnos y los contenidos de la enseñanza. Producto de la revisión bibliográfica se ha encontrado una gran variedad de recursos que pueden ser usados para enseñar y evaluar contenidos de programación, incluyendo desde grandes entornos de programación a herramientas básicas implementadas como simples páginas Web. A continuación se describen algunas de las consideradas más relevantes.

4.1 Herramientas Ejercitativas y de Autoevaluación

Existen aplicaciones que permiten la creación de herramientas ejercitativas, dando la posibilidad en algunos casos de ser trabajadas online u offline, y pudiéndose incorporar a plataformas existentes mediante la opción Scorm. Algunas de ellas son SIETTE[3], Hotpotatoes, Ardora, Exe entre otros.

Resulta primordial brindar la posibilidad de personalizar y adaptar los ejercicios en función del nivel de conocimiento de cada alumno. De esta manera si un alumno acierta una pregunta puede continuar con la siguiente pregunta o decidir finalizar el test, en consecuencia la longitud del mismo y las preguntas diferirán por cada alumno.

Se han diseñado una serie de instrumentos para la evaluación de aprendizajes basado en pruebas objetivas [8], con el fin de posibilitar al alumno autoevaluar sus conocimientos sobre el diseño de algoritmos en las metodologías diagramas de Llaves, Pseudocódigo y el lenguaje Java. También pueden ser utilizados por los docentes para evaluar algún tema específico. Los elementos de las pruebas objetivas desarrolladas incluyeron opciones de: Respuesta breve o del tipo afirmativo, Opciones verdadero/falso, Opciones múltiples, Relación o correspondencia y Completar frases. Por ejemplo este último se utilizó principalmente para el ingreso de valores correspondientes a trazas numéricas y completar fragmentos de código de programas.

[3] SIETTE (Intelligent Evaluation System usingTest for for TeleEducation). http://www.lcc.uma.es:8080/repository/fileDownloader?rfname=LCC1380.pdf

Se tuvo precaución para evitar los casos en que se admita más de una respuesta válida. Se fijaron opciones para que los docentes puedan obtener información sobre la hora de realización del ejercicio, tiempo insumido, e intentos realizados. En algunos ejercicios se limitó el número de intentos posibles permitidos. Por cada actividad el despliegue de ítems en cada intento es de modo aleatorio, en los ejercicios de arrastre se han limitado a tres los elementos a mostrar, por una cuestión de facilitar la tarea al alumno y debido a que se ha comprobado que algunos navegadores no dejan visible la barra de desplazamiento, lo que complica la realización de esta actividad. Se guarda información sobre las puntuaciones de los ejercicios, lo que permite al docente identificar rápidamente los alumnos que han completado las actividades, diferenciándolos de los que sólo las navegan. No se han especificado requisitos, esto dependerá del uso que haga el docente. En todos los ejercicios se ofrecen "pistas" que ayudan al estudiante en su resolución, en algunos casos puntuales se especificó una disminución de la calificación final cuando se las utiliza.

4.2 Plataformas Educativas

Actualmente las instituciones educativas cuentan con plataformas que permiten adicionar soporte de contenidos y canales de comunicación a los utilizados tradicionalmente. En el caso de la enseñanza de programación, el uso de los recursos provistos en estas plataformas permite compartir pequeños fragmentos de código fuente entre alumnos, profesores o tutores a través de opciones tales como mensajería, foros, wiki o email.

Otro beneficio son los indicadores de participación e interacción que facilitan el seguimiento de los alumnos. La experiencia demuestra año a año, que a pesar de que contamos con nativos digitales, muchos estudiantes tienen dificultades relacionadas con las habilidades y destrezas requeridas para manejarse en entornos virtuales. Algunas de las plataformas más utilizadas son Moodle, WebCT, WebInfo, Atutor, entre otras.

4.3 Colecciones de Ejercicios

Existen colecciones, correctores y generadores automáticos de ejercicios aplicados a problemas de programación. Se destaca como ventajas el acceso a grandes cantidades de información, intercambio de material docente, reducción de errores humanos en la corrección y mayor uniformidad en la evaluación de problemas. Además se ayuda a la detección de plagios y al apoyo a la enseñanza individualizada de los alumnos, ofreciéndoles opciones de autoevaluación de acuerdo a su nivel inicial. Como mejoras brindadas al docente pueden mencionarse la recopilación de estadísticas acerca los resultados obtenidos por los estudiantes y un potencial ahorro del tiempo destinado a la corrección. Los ejercicios presentados a los estudiantes son ejemplos, prácticas, visualización de conceptos o algoritmos, donde las metodologías, estrategias, modos de implementación e incluso a veces arquitecturas suelen ser diferentes en función del sistema desarrollado.

Algunas herramientas permiten realizar correcciones automáticas, posibilitando liberar a los profesores de una carga tediosa, además de reducir el índice de fallos humanos y ofrecer al alumno una respuesta rápida. Aunque esto no siempre es posible cuando se trabaja con algoritmos, donde se presume que cada alumno realiza una solución única. Las aplicaciones se diferencian principalmente en si la colección de ejercicios es cerrada o abierta y la posibilidad de emisión de información estadística en base a los resultados obtenidos.

Existen en la actualidad variados sistemas que pueden utilizarse para ayudar a los alumnos a aprender la sintaxis de un lenguaje determinado, uno de ellos es CodeLab[4]. Este es un sistema interactivo disponible en Internet que posee más de 300 ejercicios cortos, cada uno centrado en una idea de programación o construcción del lenguaje. Se utiliza desde el año 2002 en clases que introducen a la programación en lenguajes tales como Python, Java, C++ y C. Una de sus desventajas es que puede trabajarse en idiomas inglés o chino. Se remarcan los posibles errores en un color que puede ser diferenciado rápidamente, además de emitir mensajes relacionados a la sintaxis o ayudas para corregir el problema. Al finalizar la actividad se presenta un reporte con un resumen de lo realizado.

4.4 Uso de Pizarras y Conferencias Online

Diferentes experiencias sobre tutorías de programación realizadas mediante el empleo de pizarras y conferencias online han mostrado a estos recursos como complemento ideal, entre las clases y/o laboratorios presenciales y las herramientas de comunicación asíncronas normalmente utilizadas en la atención del alumno.

Una dificultad observada es encontrar una franja horaria coincidente para alumnos y tutores, pero como ventaja se pueden grabar las sesiones y dejarlas disponibles para otros alumnos en el futuro. Las ventajas ofrecidas están relacionadas a un aumento de cercanía entre tutor y alumno, dado que no se limita a la escritura de un texto sino que se cuenta además con posibilidades de interacción a través del uso de audio y video. Un problema detectado es que algunas pizarras no disponen de símbolos especiales en su editor, tales como sumatorias, exponentes, etc.

4.5 Entornos de Programación

El lenguaje de programación elegido debe proveer un marco conceptual que permita expresar algoritmos con claridad y simplicidad, poseer una sintaxis que sea fácil de escribir, leer e interpretar, soportar abstracción, posibilitar una rápida verificación y sobretodo ser portable, es decir independiente de las características de una máquina particular.

El paradigma Imperativo es uno de los priorizados en la enseñanza de programación en los primeros años de carreras informáticas universitarias. Los lenguajes más usados son Pascal (en nuestra Institución se uso por más de dos décadas), C, Java y Python. En cuanto a los entornos de desarrollo integrado los más

[4] CodeLab: http://codelab1.turingscraft.com/cordelab/jsp/core.jsp

usados son Eclipse y Netbeans. Si se requiere realizar trabajos colaborativos existen opciones tales como EclipseGavab, entorno de desarrollo integrado, multiplataforma, con características colaborativas. Ofrece mensajería instantánea y edición compartida de código, funcionalidades que permiten que el profesor supervise el trabajo de forma telemática y que los alumnos colaboren virtualmente. Incluye editores con resaltado de sintaxis y marcación de errores de compilación en el propio código, depuración, ayuda integrada, entre otras características. Incorpora un cliente del sistema de control de versiones denominado Subversión que permite utilizar repositorios de código para compartir programas [9].

4.6 Objetos de Aprendizaje

Existen objetos de aprendizaje diseñados con el fin de ayudar a los nuevos programadores de ordenadores, por lo general se persigue explicar un concepto en particular de programación (en un lenguaje de programación seleccionado) a través del uso de ejemplos. Un problema de reusar objetos de aprendizajes disponibles es la falta de personalización de la mayoría de ellos, por ejemplo la posibilidad de elegir un idioma conocido.

Una aplicación web interesante es CoEDApplets[5], la misma está orientada a la enseñanza y el aprendizaje de la programación. Se basa en el uso de la tecnología Applets Java, y sus principales características son el trabajo con animación para la visualización de las trazas de algoritmos y estructuras de datos. Se presenta como un portal de tipo cooperativo en el que distintos profesionales de la enseñanza pueden incorporar dinámicamente Applets que persigan el mismo objetivo, contribuyendo no sólo con códigos sino con su traducción a diversos idiomas.

5 Conclusiones

Los estudiantes presentan diferentes estilos de aprendizaje, en el caso de programación, algunos tienden a enfocarse en datos y algoritmos; otros con teorías y modelos matemáticos. Determinado grupo responde fuertemente a formas visuales de información, como gráficos, diagramas y esquemas; otros a explicaciones verbales ya sean escritos u orales. Los materiales educativos e instrumentos de evaluación deberían adaptarse al grupo destinatario, por esto resulta esencial contar con opciones que permitan flexibilizarlos o personalizarlos, de esta manera se puede respetar ritmos y diferencias individuales.

En relación a herramientas TIC para la enseñanza de programación, se destaca el uso de bancos de datos relacionados que permiten ofrecer a alumnos respuestas y correcciones de una manera más rápida y eficiente. Variados repositorios de objetos de aprendizaje permiten trabajar diversos contenidos de programación, disminuyendo los tiempos que estos insumirían al docente si tuviera que desarrollar este material en el inicio de cada curso. Las opciones como pizarras online o herramientas de videconferencia, que se adicionan a los recursos de comunicación y colaboración

[5] Coedapllets: http://www.pcg.ull.es/coedapplets/dinamico/appletList/page2/

generalmente provistos en las plataformas educativas más utilizadas, ayudan a enriquecer la interacción de los distintos participantes. Las colecciones de ejercicios constituyen un depósito útil de problemas adecuados para el conocimiento de una materia. Se pueden utilizar para realizar una labor de reconocimiento de los conceptos, evaluar conocimientos avanzados, prácticas previas a instancias evaluativas, etc.

Los ejercicios de respuesta cerrada permiten una corrección automática e inmediata, dado que sólo se compara la respuesta dada con la respuesta correcta almacenada. Si bien pueden diseñarse opciones de ejercicios que requieran completar una parte pequeña de un código de programa dado, no resultan adecuados cuando se necesita que los alumnos diseñen un programa completo. Allí intervienen elementos tales como analizadores léxicos y sintácticos, comparadores de estructuras y manejo de errores. Para estas situaciones el entorno ECDIA resulta ideal, dado que el docente no estaría limitado sólo a observar la solución final (diseño del algoritmo, codificación o ambos) sino que podría contemplar además los pasos o acciones realizados por el alumno.

Las competencias ponen de manifiesto el interés que despierta en las personas el poder responder preguntas y poner a prueba su conocimiento, por eso consideramos las competencias de programación como un recurso muy promisorio para favorecer un incremento de la motivación en la enseñanza y el aprendizaje de programación.

Referencias

1. 21stcenturyskills. Logros indispensables para los estudiantes del siglo XXI, (2004), Recuperado el 10/07/2014 de http://eduteka.org/SeisElementos.phpcompetencias
2. Fracchia, C., Martins, A.: Uso de Recursos Tecnológicos y de Técnicas de Dinámica Grupal en Materias de Programación. Encuentro de Académicos e Investigadores Patagónicos de Argentina y Chile. Punta Arenas. Chile, (2005).
3. Gonzalez, A.H., Madoz, M. C.: Utilización de TIC para el Desarrollo de Actividades Colaborativas para la Enseñanza de la Programación. TE&ET VIII Congreso de Tecnología en Educación y Educación en Tecnología, Santiago del Estero, (2013).
4. García Sierra, J., Rodríguez Lera, F., Fernández, C., Matellán Olivera, V.: Uso de Robots y Animales como Herramientas Motivadoras en la Enseñanza de Materias TIC .VAEP-RITA Vol. 1, Núm. 4, Dic. 2013 pp..203--210, (2013)
5. López García, J.C.: Educación Básica Algoritmos y Programación. (Guía para Docentes), 2° ed., (2009)
6. Levine Gutierrez, G.: Introducción a la computación y a la programación estructurada. McGraw-Hill. México. ISBN 968-451-608-8, (1985).
7. Fracchia, C., Baeza, N., Martins, A.: ECDIA: Entorno Colaborativo para el Diseño e Implementacion de Algoritmos. XX Workshop Tecnología Informática aplicada en Educación. Cacic 2012, UNS (2012)
8. Fracchia, C., Martins, A.: Diseño de pruebas objetivas para la evaluación de aprendizajes de técnicas de programación estructurada., VI Congreso Nacional y IV Internacional De Investigación Educativa "La Investigación Educativa En El Contexto Latinoamericano". Universidad Nacional Del Comahue (2013)
9. Gallego, M., Gortázar, F.: EclipseGavab, un entorno de desarrollo para la docencia online de la programación, XV JENUI. ISBN: 978-84-692-2758-9,(2009). Recuperado el 19/07/2014 de http://jenui2009.fib.upc.edu/