Entorno web como herramienta de apoyo para el diseño de algoritmos en pseudocódigo durante el proceso enseñanza-aprendizaje

Mónica A. Carreño León[1], J. Andrés Sandoval Bringas[1], Italia Estrada Cota[1], Jesús Hernández Cosío[1], Arturo de Casso Verdugo[1], Israel Durán Encinas[1]

[1]Departamento Académico de Sistemas Computacionales – Universidad Autónoma de Baja California Sur (UABCS)
Carretera al Sur Km. 5.5 – La Paz – BCS – México

{mcarreno,sandoval,iestrada,jhernandez,adecasso,iduran}@uabcs.mx

Resumen. *En este artículo se presenta el desarrollo de una herramienta didáctica que proporciona al alumno un entorno web de programación para el diseño de algoritmos básicos en pseudocódigo. A través de la herramienta se pueden diseñar algoritmos en pseudocódigo en español, comprobar su funcionamiento y desempeño, paso a paso, de manera visual, facilitando el estudio de la lógica para la resolución de problemas. Para el profesor, la herramienta proporciona un medio para monitorear el avance individual y grupal de sus alumnos, como apoyo a sus actividades dentro del curso. También se presentan algunos ejemplos, así como la experiencia obtenida en su utilización en un curso básico de introducción a la programación.*

1. Introducción

Aprender a programar se considera básico para alumnos que se encuentran estudiando una carrera en las áreas de informática y computación. El aprendizaje de las materias de programación es de los más difíciles y complejos [Arévalo & Solano 2013][Soler & Lezcano 2009]. Los alumnos han mostrado desde siempre dificultad para asimilar nociones abstractas, y por consiguiente se muestran altos índices de reprobación y de deserción en los cursos [Checa 2011]. Para aprender a programar es necesario conocer las estructuras de programación y fundamentalmente resolver muchos ejercicios: "Para aprender a programar, hay que programar" [Sánchez-Ledesma et al. 2013].

La experiencia obtenida en la enseñanza de la programación a nivel licenciatura, ha permitido detectar la dificultad a la que se enfrenta el alumno para relacionar la abstracción que requiere el diseño de un algoritmo en pseudocódigo y la obtención de los resultados esperados, es decir, para los alumnos es difícil determinar si el diseño del algoritmo es correcto y responde a los requisitos. A esto se adiciona que la solución a un problema puede ser expresado de diversas maneras, y estar bien.

En este artículo se presenta el desarrollo de una herramienta web interactiva como apoyo para el diseño de algoritmos en pseudocódigo durante el proceso de enseñanza - aprendizaje. La herramienta proporciona al alumno un entorno web de programación en el cual puede diseñar algoritmos en pseudocódigo, y comprobar su

funcionamiento y desempeño de manera visual, facilitando el estudio de la lógica para la resolución de problemas. A través de la herramienta el profesor podrá monitorear el desempeño de cada uno de los alumnos, tanto en un ambiente formal, como es el aula de clases, como en un ambiente informal, fuera del aula de clases. El artículo está estructurado de la siguiente manera: en la sección 2 se presenta el marco teórico, en la sección 3 la metodología para el desarrollo de la herramienta, en la sección 4 se detalla el funcionamiento de la herramienta, en la sección 5 se presenta la experiencia en el aula de clases de la herramienta, y finalmente en la sección 6 se muestran conclusiones.

2. Marco Teórico

2.1 Problemática de la enseñanza de la programación.

Escribir un programa de computadora utilizando un lenguaje de programación requiere de varias competencias y habilidades, que involucran básicamente la capacidad de manipular un conjunto de abstracciones interrelacionadas entre sí para la resolución de problemas [Chesñevar 2000]

La resolución de problemas implica, generalmente, un esfuerzo intelectual y el diseño de un conjunto de pasos que permitan llegar a la solución del problema [Willing et al. 2010]. En [Polya 1965] se plantea como estrategia el método de los cuatro pasos para la resolución de problemas: entender un problema, diseñar un plan, ejecutar un plan y mirar hacia atrás.

La programación es una actividad que requiere el uso de ambos lados del cerebro. Se requiere razonamiento lógico-verbal para diseñar e implementar software correctamente [Naps 2002]. En [González, Estrada & Martínez 2006] se propone que los primeros conceptos de programación se enseñen a partir de la construcción de algoritmos, usando pseudocódigos. Por otro lado en [Moroni y Señas 2005] se afirma que la complejidad de la programación hace necesario la utilización de técnicas efectivas de programación, para lo que debe ponerse especial atención en el diseño previo, por lo que se recomienda la utilización de algoritmos como recursos esquemáticos para plasmar el modelo de la resolución de un problema.

Diseñar un algoritmo semánticamente correcto no se considera una tarea trivial, es necesario llevar a cabo refinamientos sucesivos hasta ponerlo a punto. Este proceso operativamente es complicado cuando se trabaja de la manera tradicional utilizando lápiz y papel, y se intenta comprobar la correctitud del algoritmo [Moroni & Señas 2005]. Es difícil, mental o gráficamente, llevar a cabo las acciones del algoritmo en ejecución de manera totalmente objetiva sin dejarse llevar por la subjetividad de su especificación. Para ello es importante contar con un editor de algoritmos que ayude al alumno en la especificación del mismo, y que además permita comprobar su correctitud.

Esto ha generado el desarrollo de herramientas visuales para estructuras de datos y algoritmos, las cuales permiten que un alumno procese los conceptos en forma paralela, visual y lógicamente, ayudando a la asimilación de los conceptos [Naps 2002]. Existen muchas herramientas para la visualización de algoritmos. Estas herramientas se han estado desarrollando desde principios de 1980. Sin embargo estas herramientas generalmente solo son animaciones del funcionamiento de algoritmos y no permiten la interacción del usuario [Naps 2002].

Por otro lado, se pueden encontrar también herramientas de software como apoyo didáctico para facilitar la enseñanza-aprendizaje de algoritmos. En [Arellano, 2012] se menciona que entre estas herramientas se encuentran las que explotan el uso de micromundos tales como Alice y JKarelRobot. Entre las herramientas basadas en representaciones de pseudocódigo o diagramas de flujo destacan: PSeInt, RAPTOR y DFD, siendo PSeInt la más utilizada. También en [Arellano 2012] se mencionan algunos de los inconvenientes que presenta la herramienta PSeint entre ellos, no visualiza el cambio de valor de las variables, no presenta una traza completa de la ejecución, es decir, no indica la secuencia de instrucciones de su ejecución, así como el valor de las variables del algoritmo después de cada acción.

2.2 Software educativo.

Se define como software educativo a "los programas de computación realizados con la finalidad de ser utilizados como facilitadores del proceso de enseñanza" [Marques 1995] y consecuentemente del aprendizaje, con cinco características esenciales: finalidad didáctica, utilización de la computadora, interactividad, individualización del trabajo y facilidad de uso.

El proceso de elaboración de software educativo no es un proceso lineal, sino iterativo: en determinados momentos de la realización se comprueba el funcionamiento, el resultado, se evalúa el producto y frecuentemente se detecta la conveniencia de introducir cambios [Marqués 1995]. Seleccionar el ciclo de vida prototipo evolutivo permite que los expertos educativos y educadores participen de una manera más abierta y directa, involucrándose de esta manera en el desarrollo del software, lo que obviamente representa una gran ventaja, ya que ellos son los que conocen las necesidades y problemáticas de los alumnos.

La diversidad tecnológica actual, ha hecho surgir formas de integrar distintas tecnologías, dando origen a Internet, en especial world wide web (WWW) [Madueño 2003]. Los entornos de enseñanza - aprendizaje son factibles de implementar bajo plataforma web, mediante los distintos servicios que Internet ofrece a los diseñadores presentados en diferentes herramientas que facilitan que los alumnos puedan compartir, trabajar y discutir de forma sincrónica o asincrónica. De tal manera que el desarrollo de software educativo bajo plataforma web, como apoyo didáctico a sistemas tanto presenciales como a distancia, se hace necesario para lograr el mejor uso posible de las nuevas tecnologías de la comunicación y la información.

3. Metodología

Para la construcción de la herramienta se adoptó el modelo de ciclo de vida denominado prototipo evolutivo. El prototipo evolutivo se basa en la idea de desarrollar una implementación inicial exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado, permitiendo responder rápidamente a los cambios que se puedan presentar.

Durante el desarrollo de la herramienta se llevaron a cabo reuniones con profesores del área de programación e ingeniería del software. La principal característica que se consideró para el diseño de la herramienta fue considerar que es el propio alumno quien construye su conocimiento en un proceso en el cual el profesor debe actuar como

guía. En dicho proceso el alumno se enfrenta a instancias de problemas que intenta resolver hasta lograr el diseño de una solución adecuada. Se inició diseñando un prototipo que se fue refinando y ampliando hasta que el prototipo se terminó. Las etapas que se llevaron a cabo para el desarrollo de la herramienta fueron:

3.1 Identificación de requerimientos.

Entre los requerimientos identificados se encuentran:

1. Escribir algoritmos en pseudocódigo utilizando estructuras básicas.
2. Verificar la sintaxis del pseudocódigo.
3. Visualizar gráficamente la asignación de espacio de memoria para cada una de las variables según el tipo de dato.
4. Visualizar gráficamente la representación de arreglos.
5. Mostrar la salida del algoritmo por consola.
6. Verificar el funcionamiento del algoritmo paso por paso.

En la figura 1 se muestra el diagrama de casos de uso de la herramienta.

Los actores que interactúan con la herramienta son:

1. Profesor, es la persona encargada de generar prácticas y verificar el desempeño del alumno y generar las estadísticas de evaluación.
2. Alumno, es el usuario principal de la herramienta.

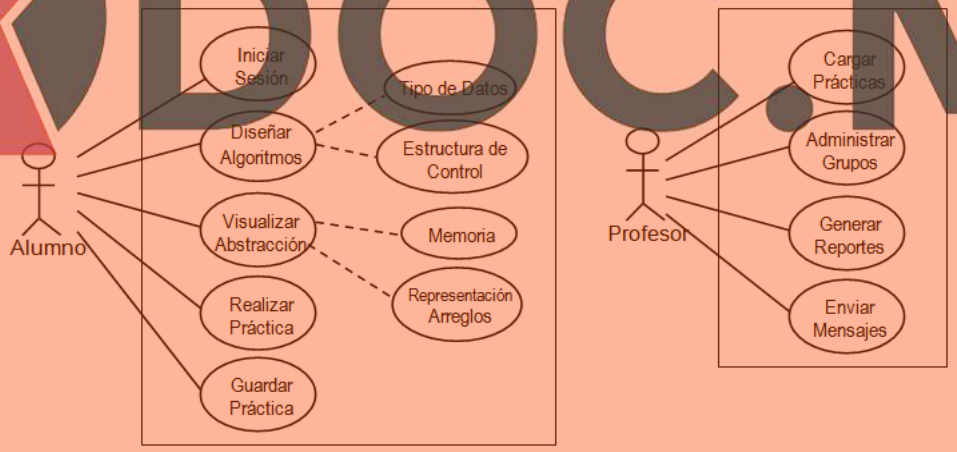

Fig. 1. Diagrama de casos de uso de la herramienta didáctica.

Los casos de usos de la herramienta son:

1) *Iniciar Sesión*: Permite ingresar a la herramienta mediante el usuario y contraseña registrados al crear una cuenta.
2) *Diseñar Algoritmos*: Permite construir algoritmos utilizando tipos de datos y estructuras básicas de control.
3) *Visualizar Abstracción*: Permite mostrar una abstracción del contenido de la memoria y el funcionamiento de una operación en particular.
4) *Verificar Funcionamiento*: Permite verificar el funcionamiento del algoritmo línea por línea durante la ejecución.
5) *Realizar Práctica*: Permite llevar a cabo las prácticas que se tienen cargadas en la herramienta.

6) *Cargar Prácticas*: Permite agregar la definición de ejercicios como prácticas.
7) *Administrar Grupos*: Permite realizar las acciones de registro y consulta de alumnos a través de grupos para facilitar la manipulación de los logros.
8) *Generar Reportes*: Permite realizar evaluaciones y consultarlas para observar el nivel de avance del alumno como ha sido su proceso de aprendizaje a través del uso de la herramienta.
9) *Enviar mensajes*: Permite al profesor enviar mensajes individuales o de grupo.

3.2 Diseño, desarrollo e implementación del prototipo.

En la figura 2 se muestra el diagrama general de la herramienta en donde se puede apreciar los módulos que la conforman, y la interacción entre ellos, bajo el patrón de diseño Modelo–Vista–Controlador (Model–View–Controller, MVC por sus siglas en inglés). El MVC permite separar la interfaz gráfica de usuario, del modelo de datos y de la lógica; este patrón se utiliza frecuentemente en aplicaciones Web.

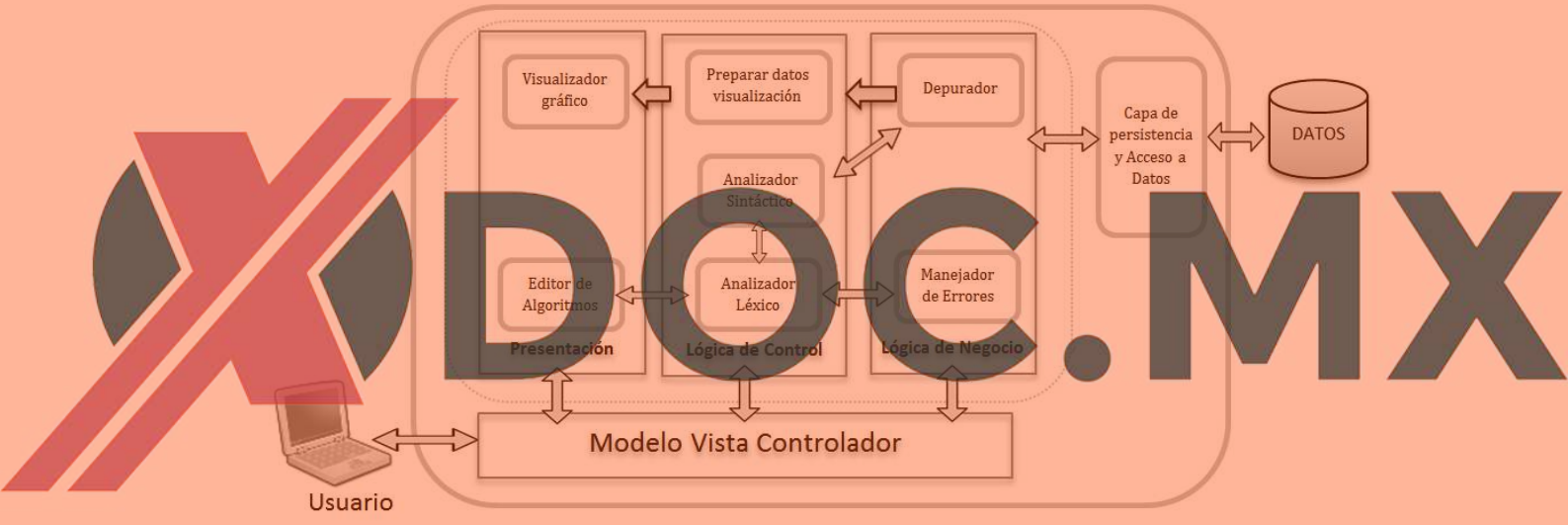

Fig. 2. Diagrama general de la herramienta.

1) *Editor de algoritmos*. Permite introducir instrucciones en pseudocódigo.
2) *Analizador léxico*. Permite leer caracteres para formar componentes e identificarlos o clasificarlos y pasar la información al analizador sintáctico.
3) *Manejador de errores*. Permite identificar los errores del analizador léxico y el analizador sintáctico.
4) *Analizador sintáctico*. Permite verificar que las instrucciones introducidas sean válidas y se encuentren escritas correctamente conforme a la gramática predefinida.
5) *Depurador*. Permite la ejecución paso a paso del pseudocódigo.
6) *Preparar datos para visualización*. Genera el código fuente en el lenguaje de programación C#, a partir del pseudocódigo.
7) *Visualizador gráfico*. Ejecuta internamente el código fuente y visualiza los resultados de la ejecución utilizando la representación gráfica.

La herramienta se implementó utilizando para el desarrollo diferentes tecnologías y herramientas, PHP como lenguaje del lado del servidor para el procesamiento de las páginas. Las páginas se codificaron haciendo uso de HTML5.

Javascript se utilizó en el lado del cliente para el manejo de llamadas asíncronas al servidor por medio de AJAX, también para el manejo de la interfaz gráfica de la herramienta, para la generación del analizador léxico y sintáctico, así mismo para la ejecución del código. El programa que se codifica finalmente es convertido a código de Javascript y es ejecutado por el motor de Javascript. Para la generación del analizador léxico y sintáctico (parser) se utilizó la librería Jison de Javascript.

3.3 Prueba del prototipo y refinamiento iterativo.

Durante el tiempo del desarrollo de la herramienta fue posible la interacción directa con profesores del área de programación e ingeniería del software, así como con expertos educativos. Ellos realizaron pruebas al prototipo de manera directa, haciendo observaciones y recomendaciones, las cuales se fueron incorporando y haciendo el refinamiento iterativo para alcanzar el prototipo final deseable.

4. Descripción de la herramienta

La herramienta proporciona al alumno un entorno web de programación en el cual puede diseñar algoritmos básicos en pseudocódigo, comprobar su funcionamiento y desempeño de manera visual, utilizando estructuras de control básicas: de ejecución secuencial, de selección e iterativas. La figura 3 muestra el área de trabajo de la herramienta, la cual se encuentra dividida en cinco secciones: panel para pseudocódigo, panel de visualización de consola, panel de opciones, panel de representación de la memoria y panel para la representación gráfica de arreglos.

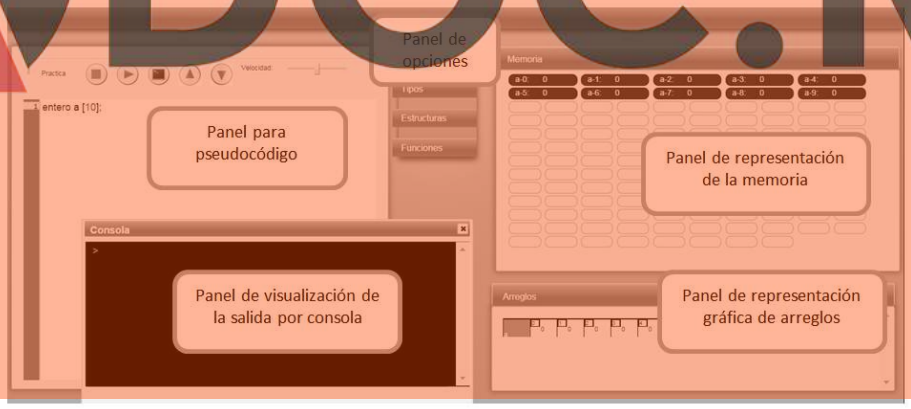

Fig. 3. Área de trabajo de la herramienta.

Panel para pseudocódigo.- En esta sección el usuario puede introducir directamente las instrucciones en pseudocódigo para ejecutarlas posteriormente. La herramienta contiene un analizador sintáctico que verifica cada una de las instrucciones antes de ejecutarlas. En este panel se encuentra una barra de herramientas con botones que permiten ejecutar, detener, pausar e incrementar o disminuir la velocidad de ejecución del algoritmo.

Panel de visualización de consola.- En esta sección se muestra la salida por consola durante la ejecución del pseudocódigo, en caso de que el algoritmo lo contemple.

Panel de opciones.- En esta sección se encuentran las opciones que el usuario puede seleccionar: Archivo, Tipos, Estructuras y Funciones.

Panel de representación de la memoria.- En esta sección se muestra una representación gráfica de la memoria para el almacenamiento de las variables utilizadas.

Panel de representación gráfica de arreglos.- En esta sección se muestra una representación gráfica de los arreglos definidos dentro del pseudocódigo.

Las instrucciones en pseudocódigo permitidas en el editor de algoritmos de la herramienta se muestran en la figura 4. Es necesario hacer la declaración de variables que se utilizarán. Los tipos de datos permitidos son: entero, flotante, lógico, cadena y arreglo. La sintaxis para la declaración de variables es: *tipo_de_dato variable* ; para el caso de los arreglos es: *tipo_de_dato nombre_arreglo [dimensión_arreglo]*. Todas las instrucciones terminan con punto y coma.

Instrucción en pseudocódigo	Descripción	Sintaxis	Ejemplo
leer	Permite introducir un valor desde el teclado	variable = *leer*(tipo de dato);	x=*leer*(entero);
imprimir	Permite desplegar el contenido de una variable o el valor de una cadena	*Imprimir*(variable/cadena);	*Imprimir*(x); *Imprimir*("hola");
nl	Agrega un salto de línea	*Imprimir*("mensaje" + *nl*);	*imprimir*("hola " + *nl*); *imprimir*("mundo");

Fig. 4. Instrucciones en pseudocódigo permitidas por el editor de algoritmos.

En la figura 5 se muestra la sintaxis para las instrucciones condicionales e iterativas que se pueden utilizar en el editor de algoritmos de la herramienta.

```
si condición entonces
    // instrucciones
sino
    // instrucciones
finsi
```

```
para variable = inicio hasta fin con paso factor
    // instrucciones
finpara
```

```
Mientras condición hacer
    // instrucciones
finmientras
```

```
repetir
    // instrucciones
hasta que condición
```

Fig. 5. Sintaxis de las instrucciones condicionales e iterativas en pseudocódigo.

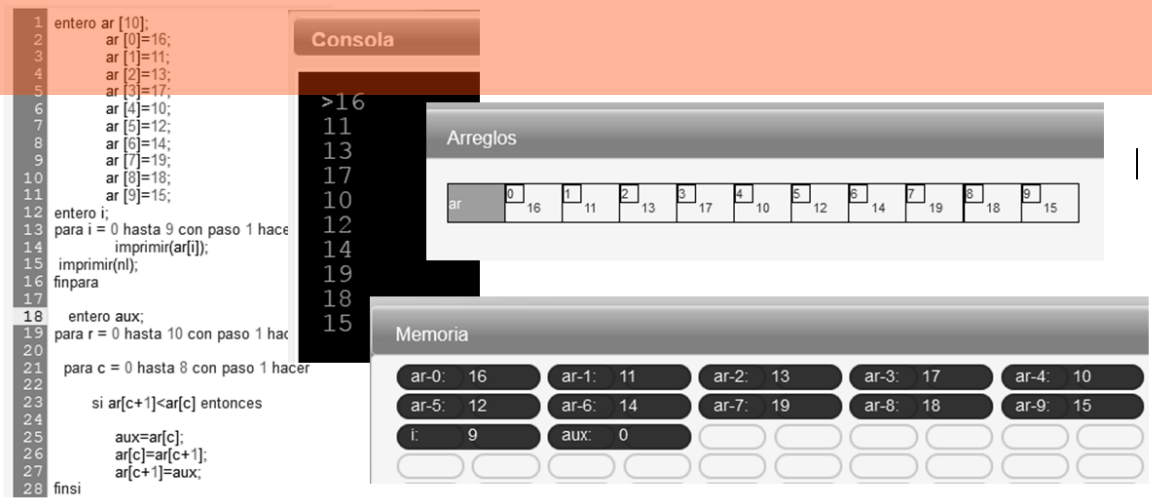

Fig. 6. Ejecución de un algoritmo en pseudocódigo y su representación gráfica.

La conexión entre el algoritmo en pseudocódigo y su representación gráfica durante el tiempo de ejecución, ocupa una parte muy importante en esta herramienta debido a la relevancia que tiene la representación gráfica de la abstracción del almacenamiento en memoria de las variables, así como del manejo de arreglos. La herramienta permite ejecutar el algoritmo en pseudocódigo paso a paso, controlando la velocidad, para dar seguimiento a los valores de las variables. En figura 6 se puede apreciar el estado de la memoria y la representación gráfica del contenido del arreglo después de ejecutar las primeras 18 líneas del pseudocódigo paso a paso. También se puede observar la salida que muestra la consola, después de ejecutar las líneas 13, 14, 15 y 16.

5. Experiencia en el aula con la herramienta

La herramienta se utilizó en un curso básico de introducción a la programación, para obtener información inicial sobre la aceptación de la herramienta y el entendimiento de la misma por parte de los alumnos. La herramienta fue utilizada por 50 alumnos durante 4 semanas.

Como primera actividad se presentó la herramienta a los alumnos, así como la sintaxis que se utiliza para el desarrollo de algoritmos en pseudocódigo. Se asignaron tres prácticas inicialmente:

1. Generar los números pares entre 0 y 100,
2. Crear un arreglo de 10 elementos aleatorios, y ordenarlos de menor a mayor, y
3. Crear un arreglo de 10 número aleatorios, y encontrar el menor, el mayor y calcular el promedio.

Las prácticas fueron desarrolladas primeramente de manera tradicional, con lápiz y papel; posteriormente utilizando la herramienta. Durante el desarrollo de cada una de las prácticas al utilizar la herramienta el alumno pudo comprobar en todo momento el diseño del algoritmo, obtener ayuda de la sintaxis de las instrucciones, así como retroalimentación en el caso de los errores. Al momento de ejecutar los pseudocódigos cada alumno pudo comprobar si el algoritmo diseñado por él, cumplía con las indicaciones de la práctica.

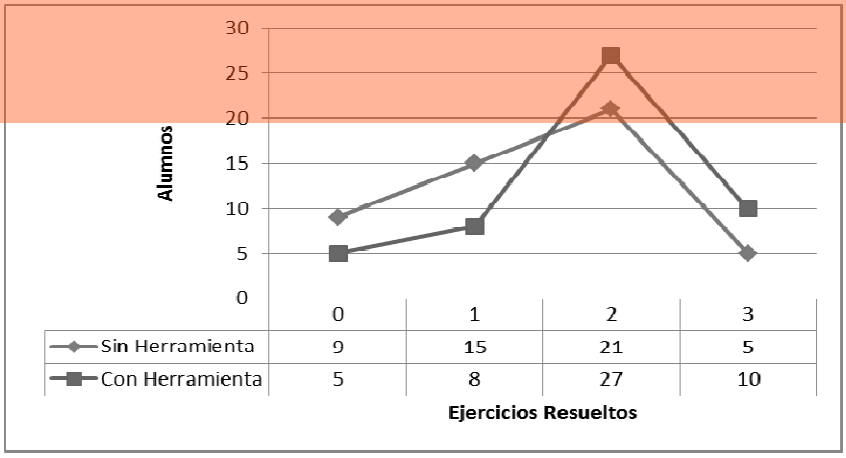

Fig. 7. Progreso de los resultados obtenidos de la práctica inicial de los alumnos.

Con la realización de estas prácticas se pudieron obtener resultados positivos. En la figura 7 se puede observar el progreso de los resultados obtenidos al utilizar la

herramienta por los alumnos. Los alumnos pudieron comprobar el funcionamiento paso a paso y el desempeño de los algoritmos de manera visual. La aceptación de la herramienta fue inmediata, hubo un gran interés por utilizarla para el desarrollo de otras prácticas. Se pudo notar que los alumnos mejoraban la habilidad para el desarrollo de algoritmos al presentar la ejecución de la misma de manera visual. Los alumnos comentaron las ventajas que representaban el desarrollar los algoritmos utilizando la herramienta. También indicaron que con esta nueva herramienta podrían desarrollar de manera más rápida las prácticas que se tenían asignadas para el tema, e incluso desarrollar nuevas prácticas. Finalmente, la herramienta fue utilizada por los alumnos para desarrollar todas las prácticas que se tenían planeadas para el curso.

6. Conclusiones

El uso de las tecnologías de la información y la comunicación representa hoy en día uno de los avances más significativos en el ámbito educativo, proporciona tanto al profesor como al alumno una útil herramienta tecnológica, provocando que el alumno se convierta en protagonista y actor de su propio aprendizaje. Los entornos de aprendizaje virtuales constituyen un recurso didáctico complementario, que debe ser empleado adecuadamente dentro de un amplio proyecto docente.

La herramienta proporciona un entorno web de aprendizaje, facilitando la enseñanza de algoritmos a través de un pseudocódigo en español, con reglas de sintaxis sencillas y básicas, permitiendo que el alumno se concentre en el estudio de la lógica para la resolución de problemas. A través de la herramienta el alumno puede detectar errores sintácticos y semánticos; visualmente puede comprobar el funcionamiento de un algoritmo verificando el contenido de las variables, con la ejecución paso a paso de cada una de las instrucciones que contiene el algoritmo; puede comprender lo que sucede con las estructuras de control. También es posible hacer animaciones del funcionamiento de los algoritmos para los métodos de ordenación, de búsqueda, entre otros.

La herramienta diseñada y desarrollada permite despertar el interés del alumno, motivarlo y mantenerlo participativo en el proceso de la enseñanza - aprendizaje de la programación. Para el profesor, la herramienta se convierte en un instrumento valioso como apoyo a sus actividades, permitiendo monitorear el avance individual y grupal de sus alumnos.

Referencias

Arellano, J., Nieva, O., Solar, R. and Arista, G. (2012). Software para la enseñanza-aprendizaje de algoritmos estructurados. In *Revista Iberoamericana de Educación en Tecnología y Tecnología en Educación*, páginas 23–33. Universidad Nacional de La Plata (RedUNCI – UNLP).

Arévalo, C. and Solano, L. (2013). Patrones de Comportamiento de Estudiantes de Programación al Utilizar una Herramienta de Visualización de Protocolos Verbales. In *8va Conferencia Latinoamericana de Objetos de Aprendizaje y Tecnologías de Aprendizaje, LACLO 2013*, páginas 1–11. Valdivia, Chile, del 21al 25 de octubre de 2013, ISSN: 1982 – 1611.

Checa, R. (2011). La innovación metodológica en la enseñanza de la programación. Una aproximación pedagógica al aprendizaje activo en la asignatura Fundamentos de

Programación. In *Interfases Revista Digital de la Facultad de Ingeniería de Sistemas*, páginas 67–87. Universidad de Lima.

Chesñevar, C. (2000) "Utilización de Mapas Conceptuales en la enseñanza de la programación", http://cs.uns.edu.ar/~cic/2000/2000-jornadas-mapas/2000-jornadas-mapas.pdf, Consultado el 10 de febrero de 2015.

Gottberg, E., Noguera, G. and Noguera, M.A. (2011). Propuesta pedagógica: Una Metodología de desarrollo de software para la enseñanza universitaria. In *UDUAL*, páginas 49–57. www.redalyc.org.

Levy, L. (1994). From Specific Problem Instances to Algorithms in the Introductory Course. In *SIGCSE Bulletin ACM*.

Madueño, L. (2003). Desarrollo de Software Educativo bajo Plataforma Web. In *Congreso Internacional EDUTEC.*

Marqués, P. and Marqués, P. (1995), Software educativo: guía de uso y metodología de diseño, Estel.

Moroni, N. and Señas, P. (2005). Un entorno de aprendizaje de la programación. In *Primeras Jornadas de Educación en Informática y TICS en Argentina.*

Naps, T et al. (2002). Exploring the Role of Visualization and Engagement in Computer Science Education. In *SIGCSE Bulletin ACM*.

Polya, G., (1965), Como plantear y resolver problemas, Trillas, México.

Salgado, A., Alonso, I., Gorina, A. and Tardo, Y. (2013). Lógica algorítmica para la resolución de problemas de programación computacional: una propuesta didáctica. In *Revista Didasc@lia: D&E*, páginas 57–76. Publicación cooperada entre CEDUT-Las Tunas y CEdEG-Granma, CUBA.

Sánchez-Ledesma, F., Ortiz, O. and Pastor, J. (2013). Aprendizaje de los lenguajes de programación en la educación universitaria a través de dispositivos móviles. In *VI Jornadas de Introducción a la Investigación de la UPCT*, páginas 100–102.

Soler, Y. and Lezcano, M. (2009). Consideraciones sobre la tecnología educativa en el proceso de enseñanza-aprendizaje. Una experiencia en la asignatura Estructura de Datos. In *Revista Iberoamericana de Educación*, páginas 1–9. Organización de Estados Iberoamericanos para la Educación, la Ciencia y la Cultura (OEI).

Willging, P., Astudillo, G., & Bast, S. (2010). Aprender a programar (¿y a pensar?) jugando. *Memorias del congreso de Tecnología en Educativa & Educación en Tecnología (TE&ET)*, 167-176.