ECDIA: Entorno Colaborativo para el Diseño e Implementación de Algoritmos

Carina Fracchia[1, 2], Natalia Baeza[1], Adair Martins[2]

[1] Departamento de Programación
[2] Departamento de Computación Aplicada,
Facultad de Informática, Universidad Nacional del Comahue
1400 Buenos Aires, Neuquén, Argentina
{cfracchi}@ hotmail.com, {nataliabaeza}@gmail.com, {adair_martins}@yahoo.com.br

Resumen. En las materias introductorias de programación es fundamental lograr que los alumnos puedan adquirir habilidades para resolver problemas mediante el diseño de algoritmos y su posterior implementación en un lenguaje determinado. Existen actualmente distintas herramientas tecnológicas que permiten el diseño de algoritmos utilizando metodologías tales como pseudocódigo y diagramas de flujo. Para la programación se usan entornos como Eclipse y NetBeans, pero los mismos carecen de funcionalidades que permitan trabajar las instancias previas correspondientes al diseño de los mismos. En este trabajo se propone el desarrollo de una herramienta basada en software libre que contempla la edición individual o colaborativa de algoritmos y al mismo tiempo su codificación en un lenguaje seleccionado como C++ o Java. La herramienta permitirá la ejecución paso a paso y la confección de la traza resultante. Será utilizada e integrada a los procesos de enseñanza y aprendizaje en los cursos iniciales de programación.

Palabras claves: Programación, Herramientas tecnológicas, Entorno colaborativo, Software libre.

1 Introducción

El presente trabajo se enmarca dentro del proyecto Simulación y Métodos Numéricos en Ciencias de la Computación de la Facultad de Informática (FAIF), Universidad Nacional del Comahue (UNCo) bajo la línea de investigación "Uso de Tecnologías de la Información y Comunicación (TIC)". El objetivo principal es el estudio y análisis de aquellas tecnologías que ofrecen potencial en la enseñanza de la programación, y que permiten la modificación y retroalimentación de los procesos de enseñanza y aprendizaje llevados a cabo.

Es muy importante conseguir que los alumnos adquirieran habilidades que les permitan resolver problemas mediante la computadora. Esto supone que los mismos a partir del enunciado de un problema pongan en práctica una serie de estrategias para el desarrollo de soluciones. Pero su tarea no termina allí, ya que debe transformar el algoritmo desarrollado en un programa escrito en un determinado lenguaje de

acuerdo a la sintaxis del mismo. Para esta tediosa tarea se utilizan distintas metodologías de diseño como pseudocódigo, diagrama de flujo entre otras [1], [2].

Existen actualmente distintas herramientas computacionales [3] a [8] que permiten en forma automática el diseño de algoritmos y su compilación en un lenguaje especificado. En el caso de los lenguajes, existen distintos software comerciales y libres. Los más utilizados son los entornos de desarrollo integrado (IDE) que proveen herramientas y funciones para la programación, desarrollo y compilación, programas en lenguajes como C++ o aplicaciones Java [9] a [11].

La programación visual para el diseño de algoritmos mediante la utilización de las herramientas mencionadas anteriormente permite que el alumno realice sus diseños eligiendo alguna metodología y a continuación pueda automáticamente compilarlo, ejecutarlo y probarlo, obviando la etapa de la codificación. La ventaja fundamental es que en esta etapa inicial no existe la necesidad de recordar la sintaxis de los lenguajes de programación utilizados en la programación convencional. El tiempo ganado se invertirá en el aprendizaje de la lógica y la verificación del correcto funcionamiento del mismo.

En este trabajo se propone el diseño de una herramienta computacional basada en software libre que integra el trabajo con metodologías para el diseño de algoritmos y lenguajes de programación tales como C++ y Java. Se contemplan las funcionalidades básicas como 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 de contar con los mecanismos de comunicación como: envío, recepción de los diseños y programas realizados.

2 Experiencia lograda en la Enseñanza de la Programación en Cursos Iniciales

La FAIF ofrece actualmente varias carreras de pregrado, grado y tecnicaturas, en las que se encuentran materias que introducen a la programación. En todas ellas se trabajan tres grandes grupos temáticos: la resolución de problemas (representación de la información y estrategias de resolución), diseño de algoritmos (empleando en la mayoría una metodología basada en Warnier-Orr y Diagrama de flujo). Para la codificación actualmente se trabajan C++ y Java) [10], [11].

La resolución de problemas y el diseño de algoritmos se realizan en papel, por lo cual los alumnos, de acuerdo a encuestas efectuadas y a observaciones realizadas en la cátedra, no detectan rápidamente los errores cometidos, o peor aún, si no consultan con la cátedra, piensan que sus diseños son correctos en situaciones que no lo son. Generalmente en la instancia en que trabajan con un lenguaje de programación, allí sumado a la tarea de lidiar con la sintaxis del mismo, descubren que la lógica empleada al resolver un algoritmo en papel no fue la adecuada, requiriendo mayor esfuerzo en las tareas de detectar y corregir errores.

La carencia de herramientas automatizadas que ayuden a los alumnos a descubrir errores en sus diseños y a validar su correctitud, es un tema pendiente y que prevalece año a año. En función de esto se han realizado diferentes experiencias, particularmente en una prueba piloto realizada en 2009 con una herramienta intérprete de pseudocódigo PSeInt [8] permitió mostrar la ventaja del uso de herramientas tecnológicas para el diseño de algoritmos. En la misma los alumnos podían intercambiar archivos con los docentes, facilitando de esta manera la corrección.

Además del beneficio de verificar sus algoritmos fuera del horario de clases. Este último fue uno de los puntos que resaltaron los alumnos en las encuestas realizadas al finalizar el cursado. Algunos de ellos como recursantes, pudieron ampliar y resaltar los beneficios comparados con dictados anteriores. Los alumnos que se ven imposibilitados de concurrir asiduamente a clases también manifestaron el beneficio de contar con una herramienta que puedan utilizar de acuerdo a sus horarios disponibles.

El intérprete de pseudocódigo utilizado pudo asistirlos en los primeros diseños realizados, facilitando la introducción de los conceptos básicos (variables, expresiones, estructuras de control, etc.) de manera independiente al lenguaje utilizado. Al disponer de ayudas básicas y asistencia se facilitó la búsqueda y solución de errores, mejorando además la comprensión de la lógica de sus diseños, y de los ejemplos brindados por los docentes. La herramienta contaba con las siguientes funcionalidades:

- Edición de algoritmos en pseudocódigo: la versión utilizada estaba en idioma español y permitía además el coloreado de sintaxis, identado automático, autocompletado, entre otros.
- Generación de diagrama de flujo del algoritmo. Esto es útil para aquellos estudiantes que traen conocimientos previos en el manejo de esta metodología.
- Edición simultánea de múltiple algoritmos.
- Ejecución de los algoritmos, dando la posibilidad de realizarlo además paso a paso, controlando la velocidad e inspeccionando las variables y expresiones.
- Resaltado de errores de sintaxis.

3 ECDIA: Diseño de la Herramienta Propuesta

Se han relevado numerosas herramientas [3] a [8] que se diferencian principalmente en las siguientes características:

- Permiten sólo la edición de algoritmos, o además la interpretación.
- Modalidad de trabajo:
    - Online, offline o ambas
    - Individual, colaborativa, ambas
- Estilos de interacción: cuadros de texto, botones, menú, íconos
- Generación de historial

En la figura 1 se puede observar algunas de las características mencionadas.

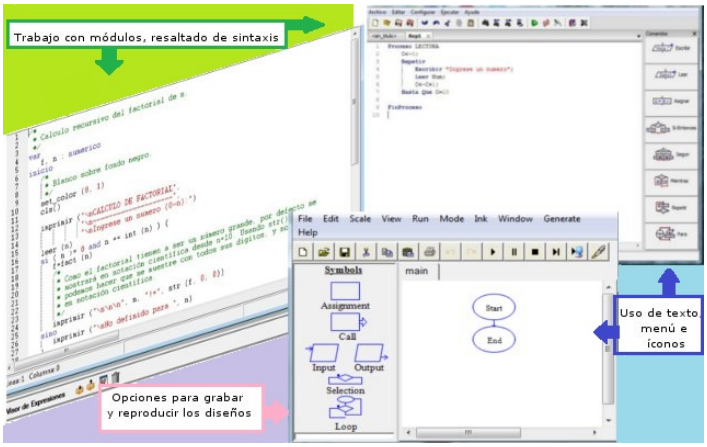

Fig.1. Algunas de las interfaces usuarias utilizadas en el diseño de algoritmos.

Las herramientas analizadas para el diseño de algoritmos son principalmente para el trabajo con pseudocódigo y diagramas de flujos. Como desventaja no poseen relación con los lenguajes de programación, algo que sería ventajoso desde el punto de vista del estudiante, que debe aprender varias herramientas en poco tiempo (esto es en relación a la experiencia descripta anteriormente). Una ventaja que resaltan algunos autores es la que los estudiantes pueden contar desde un inicio con el código correcto correspondiente a versiones del algoritmo en un lenguaje de programación seleccionado.

En función de lo descripto anteriormente se encuentra en desarrollo una herramienta basada en software libre que contempla las siguientes funcionalidades:

1. Editor gráfico de algoritmos: pseudocódigo y diagrama de flujo. Se prevé el uso de color en la sintaxis, el marcado de errores de compilación en el código, identación automática, plantillas de comandos, autocompletado y ayuda integrada de manera emergente. Se permitirá el trabajo en simultáneo con múltiple algoritmos, posibilitando que este sea individual o colaborativo.
2. Interpretación de los algoritmos (optando por alguna de las metodologías previstas: pseudocódigo o diagrama de flujo)
3. Ejecución paso a paso y confección de la traza resultante, permitiendo el control de la velocidad e inspección de variables y expresiones.
4. Determinación y marcado de errores de sintaxis. Errores en tiempo de ejecución.
5. Traducción a un lenguaje de programación: C++ y Java. Si bien se prevé en una primera instancia contemplar estos dos lenguajes, uno de los propósitos es poder gestionar el ingreso de módulos independientes para la incorporación de nuevos lenguajes.
6. Tutorial con ejemplos animados. Por ejemplo para el trabajo inicial con asignaciones un ejemplo interesante de acercar a los estudiantes es el intercambio de variables. Este tipo de ejercicio simple a la vista, en la práctica es uno de los que más cuesta entender a los alumnos.
7. Envío, recepción y almacenamiento remoto de programas. Uso de sistemas de control de versiones que faciliten el registro de las modificaciones realizadas sobre un código determinado, facilitando visualizar la evolución temporal de los cambios producidos en el mismo.

De acuerdo a lo mencionado existen varias herramientas que permiten editar pseudocódigo o diagrama de flujo. ECDIA está basada en una herramienta de las mencionadas a la cual mediante una reingeniería se le está adicionando la integración de las dos metodologías de diseño de algoritmo. El aporte potencial de la herramienta propuesta es el trabajo colaborativo en la edición de algoritmos, por lo que se está contemplando el diseño e implementación de herramientas de:

- Colaboración y comunicación: mensajería, chat, edición compartida, entre otras.
- Conferencia: trabajo con texto (chat), audio y video.
- Gestión colaborativa: calendario electrónico para fijar fechas de entregas de trabajos, seguimiento de las acciones realizadas por los grupos de trabajo (por ejemplo: en la edición de un algoritmo se guardaría información sobre los aportes individuales realizados por cada alumno, las correcciones realizadas por los docentes, etc.)

Los ítems 2, 3 y 4 mencionados anteriormente ya están contemplados en la aplicación seleccionada. Los otros ítems 5 a 7 son otros de los aportes de la herramienta en desarrollo. En el mismo entorno se brinda la posibilidad de alternar entre distintas vistas correspondientes al diseño y a la codificación de un algoritmo manteniendo un historial del desarrollo realizado. Este podrá ser usado por los alumnos para una autoevaluación y por el docente para detectar posibles fallas conceptuales, lo que le permitirá una intervención oportuna. Aprovechando las posibilidades ofrecidas por las TIC, la ayuda contemplará además de hipertexto la animación de los ejemplos entre otros.

El testeo del sistema se realizará según pruebas automatizadas y/o manuales que serán realizadas por los docentes de las materias y alumnos que cursen las materias introductorias a la programación. En la figura 2 se muestra la interfaz gráfica de la herramienta propuesta.

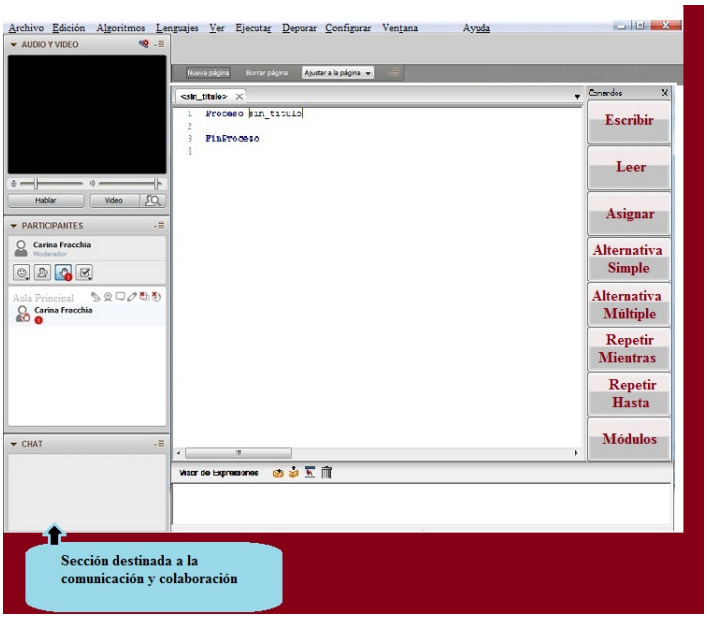

Fig. 2 Interfaz gráfica de la herramienta propuesta

A través del uso de esta herramienta en un curso, se podrán generar repositorios de diseños de algoritmos, accesibles al equipo de cátedra docente, quienes podrán detectar más fácilmente errores (manualmente o de forma automática empleando detectores específicos), facilitando las tutorías de las prácticas de laboratorio, y detectando tempranamente malas prácticas de programación. En las prácticas presenciales, los docentes no siempre tienen la posibilidad de corregir todos los diseños realizados por los alumnos.

6 Conclusiones

Los análisis realizados de las distintas herramientas automatizadas para el diseño de algoritmos han permitido observar ventajas, con su incorporación, en la enseñanza y aprendizaje en materias iniciales de programación.

Si bien existen distintas herramientas que posibilitan la edición compartida de un documento (Google Docs, Wiki, etc.), esta característica raramente se encuentra en herramientas diseñadas para el trabajo con diagramas de flujo y pseudocódigo.

El entorno ECDIA está basado en una herramienta que integra el trabajo con metodologías de diseño de algoritmo y lenguajes de programación. Su principal aporte es adicionar la posibilidad de trabajo colaborativo en la edición de algoritmos, la misma permitirá que alumnos y profesores, vean y editen el mismo diseño simultáneamente, donde mediante un cambio de rol se podrá rotar entre visualización, edición y supervisación.

Referencias

1. Criado Clavero, M.A., Programación en Lenguajes Estructurados, Alfaomega, (2006)
2. Moroni, N., Señas, P., Estrategias para la Enseñanza de la Programación, JEITICS, pp. 254--258, UNS, Bahía Blanca, (2005)
3. Garner, S., Learning Resources and Tools to Aid Novices Learn Programming, Informing Science InSITE, pp. 214--222, (2003)
4. Watts, T., The SFC editor a graphical tool for algorithm development, JSSC, USA, (2004)
5. Carliste, M.C., Wilson, T.A., Humphries, J. W., Hadfiedl, S. M., Raptor: A Visual Programiming Environment for Teaching Algorithmic Problem Solving, SIGCSE'05, pp. 23--27, USA, (2005)
6. Garner, S., A Program Design Tool Help Novices Learn Programming, Procceeding Ascilite, Singapore, (2007)
7. Manso, A., Marques, C. G., Dias, P., Portugol IDE v3.x: A New Environment to Teach and Lear Computer Programming, EDUCON, IEEE, pp. 1007--1010, (2010)
8. PSeInt, http://pseint.sourceforge.net/index.php?page=descargas.php
9. Software libre, http://www.opensource.org/
10. Entorno de programación NetBeans, http://netbeans.org
11. Entorno de programación Eclipse, http:/www. Eclipse.org