ALT: Algorithm Learning Tool

R. Laza, D. Glez-Peña, J. R. Méndez, F. Fdez-Riverola, J. Baltasar García, M. Reboiro
ESEI: Escuela Superior de Ingeniería Informática
Universidad de Vigo
Campus Universitario As Lagoas s/n, 32004 Ourense
{rlaza, dgpena, moncho.mendez, riverola, jgarcia}@uvigo.es, mrjato@correo.ei.uvigo.es

Resumen

Este trabajo presenta ALT, una herramienta on-line para la visualización gráfica, intuitiva y paso a paso de algoritmos computacionales escritos en Java. ALT es una herramienta extensible, que permite la inclusión de forma sencilla de nuevos algoritmos que se puedan representar gráficamente. Esta herramienta, desarrollada como Proyecto Fin de Carrera en el Departamento de Informática de la Universidad de Vigo, brinda un nuevo y útil recurso docente a los profesores de asignaturas de Informática donde se impartan materias relacionadas con la algoritmia.

1. Introducción y Motivación

En los planes de estudio actuales, por lo general la mayor parte de las horas lectivas se dedican a clases teóricas presenciales, que dejan menos tiempo del que profesor y alumno desearían para la realización de prácticas y/o ejercicios sobre la teoría. Además, es habitual encontrarse con que el alumno no realiza los ejercicios que se le proponen o que, teniendo dudas, no acude a tutorías para resolverlas.

Para solucionar estos problemas, los profesores de la materia MTP (*Metodología y Tecnología de la Programación*) impartida en la Escuela Superior de Ingeniería Informática (ESEI) de la Universidad de Vigo, propusieron la realización de un nuevo portal Web para la asignatura. El objetivo inicial del proyecto era el de servir de apoyo a la materia e implementar un lugar de trabajo común en el cual se pudiesen proponer distintos ejercicios, con sus correspondientes soluciones, para que el alumno los pudiese realizar o consultar de una manera más cómoda.

Partiendo de esta proposición inicial, se avanzó hasta concretar dos tipos de ejercicios que deberían estar presentes: (*i*) los cuestionarios de preguntas cortas y (*ii*) los ejercicios tipo test. Ambos tipos de ejercicios proporcionaban una solución genérica para la realización de ejercicios sobre los temas en los que no se pudieran proponer actividades más concretas.

Sin embargo, y ya desde un punto de vista más cercano a la materia MTP, es preciso destacar que los algoritmos tratados habitualmente son un elemento de difícil comprensión para los alumnos (cursando el primer año de Ingeniería Informática). Tras repasarlos en profundidad, se llegó a la conclusión de que los algoritmos de búsqueda y ordenación impartidos podían ser representados y "animados" de un modo gráfico, hecho que facilitaría en gran medida el estudio y comprensión de los mismos.

La intención de buscar nuevas formas de facilitar la enseñanza de algoritmos no es nueva [1,4,5]. En este sentido, el recurso más habitual se basa en utilizar precisamente animaciones de la ejecución de los mismos. Las diferencias existentes entre las diversas alternativas suelen estribar en el nivel de control de las animaciones (si permiten o no detener el algoritmo, avanzar paso a paso, etc.), la posibilidad de visualizar el valor de las variables o la vistosidad de la interfaz.

No obstante, la carencia principal de los animadores de algoritmos existentes radica en la extensibilidad, es decir, la posibilidad de modificar o incluir nuevos esquemas por parte de los alumnos. Este aspecto se consideró como una característica clave, ya que el poder hacer modificaciones puntuales sobre los algoritmos sería de gran ayuda para facilitar su comprensión.

Este artículo se presenta la herramienta ALT (*Algorithm Learning Tool*), como resultado final de la implementación real de esta idea. ALT constituye un sistema complejo, que permite tanto el control de la ejecución de algoritmos como su creación, modificación y posterior visualización por el alumno.

Concretamente, ALT se encuentra integrada y disponible dentro del nuevo portal Web de la

asignatura MTP (ver apartado de conclusiones), cuyos profesores planean poner en explotación en el próximo curso académico.

Mientras que la primera sección de este trabajo introduce la motivación existente de cara a la implementación de ALT, el resto del artículo se estructura como sigue. En primer lugar se detalla el funcionamiento y manejo de ALT durante la animación de los distintos algoritmos existentes. A continuación se describe cómo los alumnos pueden incluir nuevos algoritmos. Posteriormente se comentan brevemente algunos detalles de implementación y, finalmente, se exponen las conclusiones y trabajo futuro.

2. Animación de algoritmos con ALT

Esta sección describe el funcionamiento de ALT comentando, por un lado, los aspectos más destacables en la visualización y control de la animación de los algoritmos implementados: ordenación por burbuja, inserción, inserción binaria, selección, shell, MergeSort y QuickSort, además de búsqueda líneal, binaria, binaria recursiva y mediante un array índice.

2.1. Visión general

La Figura 1 muestra el aspecto de ALT durante la ejecución de un algoritmo de ordenación (ordenación por burbuja).

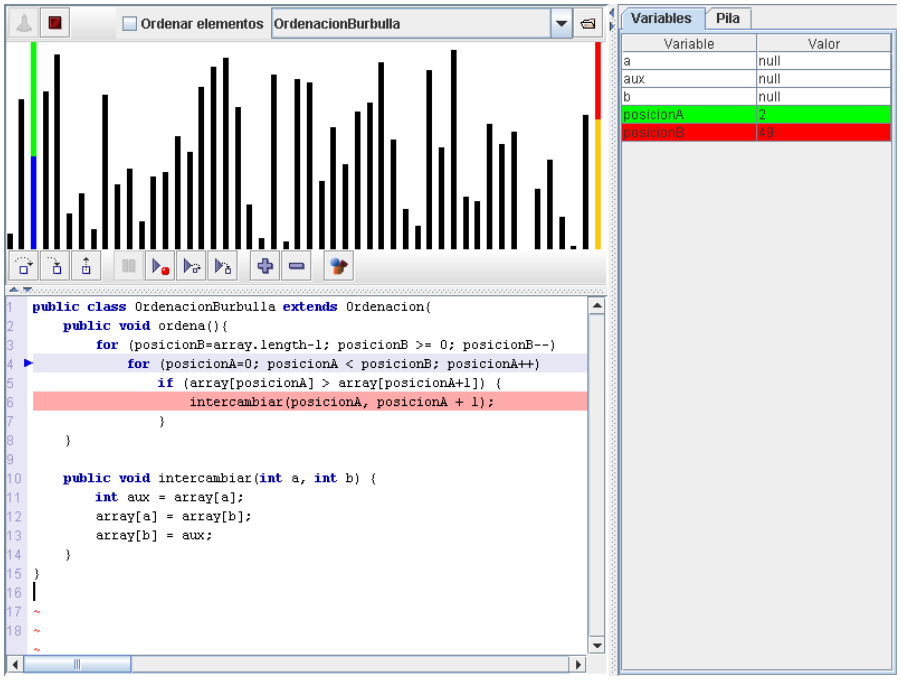

Figura 1. Aspecto general de ALT

La aplicación se puede dividir en cuatro zonas claramente diferenciadas:

- **Las barras de herramientas.** Se sitúan rodeando el visor del vector. Contienen los controles para el manejo de la animación.
- **Visor del vector**. Se sitúa en la parte superior izquierda y se encarga de mostrar el estado del vector durante la ejecución de la animación.
- **Visor del código fuente.** Se sitúa en la parte inferior izquierda y muestra el código fuente del algoritmo. Permite establecer puntos de ruptura en la ejecución y, durante la misma, resaltará la línea por donde se vaya ejecutando el algoritmo.
- **Panel de variables.** Se sitúa en la parte derecha y presenta dos pestañas: “Variables” y

"Pila". Permite visualizar los valores de las variables empleadas en el algoritmo y los registros de activación de las llamadas a funciones en la pila.

2.2. Control de la ejecución

La ejecución (o "animación") del algoritmo se maneja a través de la barra de herramientas, que contiene una serie de controles similares a los de los depuradores de los entornos de desarrollo o IDE (*Integrated Development Environment*).

Figura 2. Barra de herramientas de control

Tal y como muestra la Figura 2, el alumno puede efectuar las siguientes operaciones:

- *Step over, step into* y *step out* . Permiten saltar sobre una línea de código, entrar en un método o salir del mismo, respectivamente. Estos controles son de uso común en los IDE.
- *Pausa* . Detiene la ejecución.
- *Animar hasta breakpoint* . Continúa la ejecución hasta encontrar un punto de ruptura previamente establecido.
- *Continuar con step over* . Continúa la ejecución saltando línea a línea, pero con un pequeño intervalo de tiempo entre cada paso.
- *Continuar con step into* . Igual que el anterior, pero entrando en las llamadas a métodos.
- *Regular la velocidad* . Permite ajustar la velocidad de la animación, es decir, el intervalo de tiempo entre cada paso.

2.3. Visor del vector

El estado actual del vector sobre el que trabaja el algoritmo se puede ver en todo momento a través de un visor especializado, tal y como se muestra en la Figura 3. Dicho visor representa el vector en forma de gráfico de barras, donde el eje de abscisas representa las posiciones y el eje de ordenadas el valor del contenido en cada posición del vector.

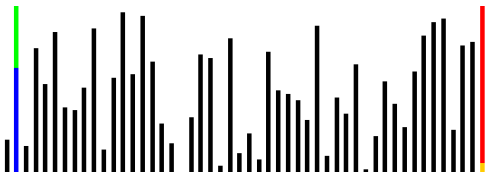

Figura 3. Panel de visualización del vector

Como se puede observar en la Figura 3, el visor permite además resaltar variables que almacenan posiciones del vector (índices) mediante dos colores. Uno de ellos cubre hasta la longitud de la barra, mientras que el otro alcanza la cima del panel con un color asociado a la propia variable que permite su identificación.

En ALT se pueden distinguir tres tipos de variables en función de su visualización:

- **Observadas.** Aparecen en el panel de variables con su valor.
- **Gráficas.** Son variables Observadas que además se les asocia un color y pueden ser vistas en el visor del vector *sólo* si se hace clic sobre ellas en el panel de variables.
- **Visibles.** Son variables Gráficas que se muestran *siempre* en el visor del vector.

Con el fin de mejorar la visualización de ciertos aspectos relevantes en la ejecución de algoritmos ALT permite la definición de **zonas.** Una zona representa una región del vector comprendida entre dos variables Visibles, que será representada en el visor del vector mediante un color de fondo diferente, tal y como muestra la Figura 4.

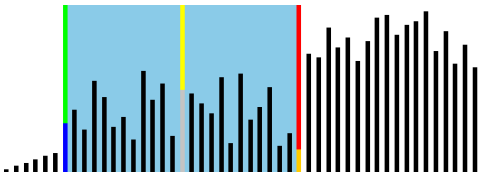

Figura 4. Zona señalada en el visor del vector

Las zonas han sido pensadas para algoritmos recursivos, donde muchas veces existen dos variables que delimitan la región del vector que se está tratando (por ejemplo en la ordenación QuickSort).

2.4. Visor de código fuente

Este visor muestra el código fuente de los algoritmos. Permite crear y eliminar *breakpoints*

en las distintas líneas que provocarán que la ejecución se detenga cuando sean alcanzadas.

La Figura 5 muestra el aspecto del visor de código fuente de un algoritmo a punto de ejecutar la línea 4 y donde se ha establecido un punto de ruptura en la línea 6.

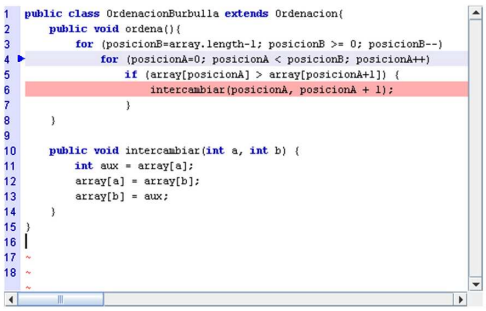

Figura 5. Visor del código fuente

2.5. Panel de variables y de pila

Este panel contiene a su vez dos componentes: una solapa para mostrar el valor de las variables y otra para visualizar la pila de ejecución.

La primera de ellas presenta una lista de variables junto con su valor y, en caso de ser índices del vector, su color asociado, tal y como muestra la Figura 6. Permite además realizar una serie de operaciones como son la selección de variables gráficas para que sean señaladas en el visor del vector, además de añadir o eliminarlas de la lista en durante la ejecución del algoritmo.

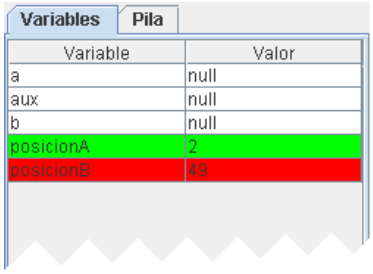

Figura 6. Panel de variables

La segunda de las solapas muestra un árbol que presenta un resumen del estado actual de la pila. El elemento principal son las llamadas a métodos, que a su vez contienen todas las variables visibles dentro de ese método. Esta vista es muy útil para comprender el funcionamiento de los algoritmos recursivos, donde se suceden llamadas al mismo método de forma continuada. La Figura 7 muestra el aspecto de esta solapa durante la ejecución del algoritmo recursivo QuickSort.

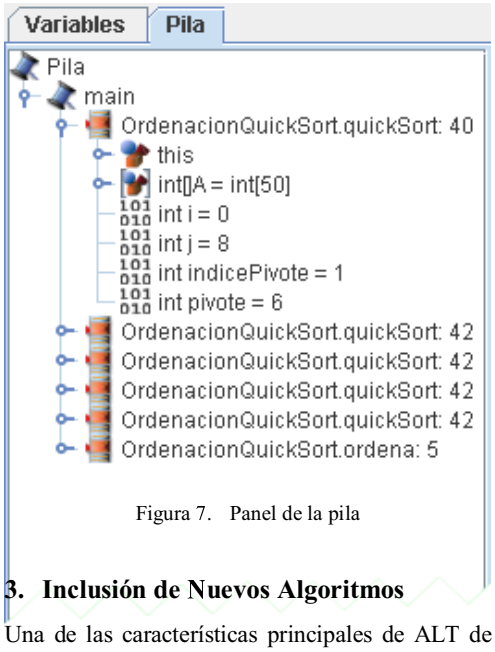

Figura 7. Panel de la pila

3. Inclusión de Nuevos Algoritmos

Una de las características principales de ALT de cara al aprendizaje, es la posibilidad de que los alumnos puedan simular la ejecución de sus propios algoritmos. Gracias a esta funcionalidad, se pueden llevar a cabo modificaciones de los algoritmos impartidos en clase y comprobar cómo afectan los cambios de una forma inmediata.

La inclusión de un nuevo algoritmo se realiza a través de la página Web de la asignatura donde se integra el sistema ALT. El alumno deberá rellenar una plantilla como la que se muestra en la Figura 8.

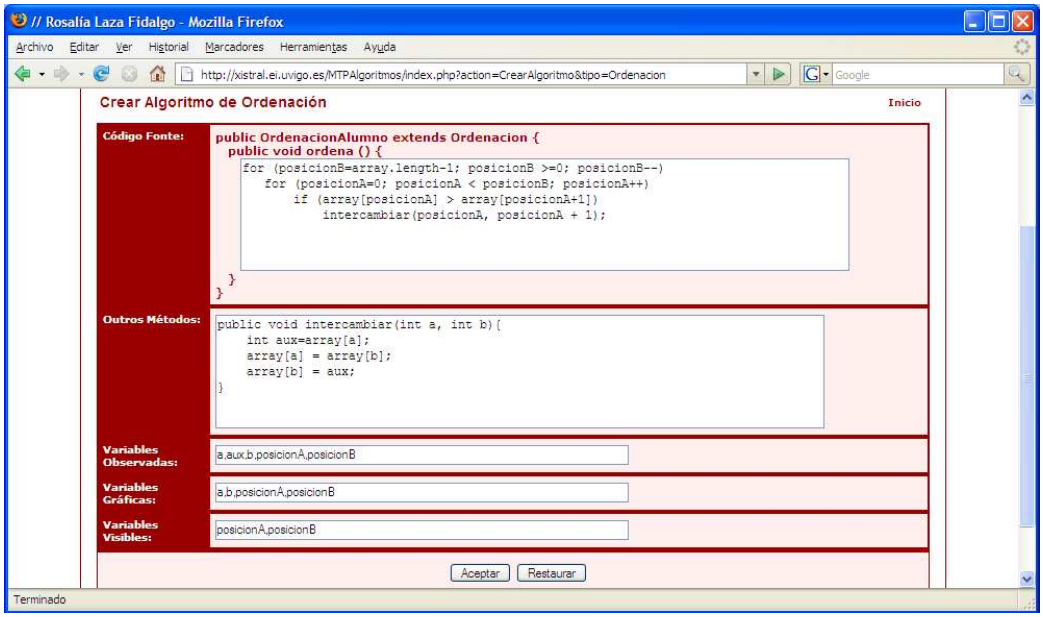

Figura 8. Formulario para la creación de nuevos algoritmos en ALT

Los campos a cubrir son los siguientes:

- **Código fuente**. Es el código del método de ordenación o búsqueda. Si el algoritmo es de búsqueda se deberá devolver un entero, mientras que si es de ordenación no se devolverá nada. Es necesario saber que el array a ordenar se encuentra en una variable denominada precisamente "array".
- **Otros métodos.** En muchas ocasiones es necesario, o más didáctico, el uso de métodos auxiliares. Para ello se proporciona otro campo donde se pueden introducir uno o varios métodos adicionales.
- **Variables observadas.** Una lista separada por comas de las variables que se desea que aparezcan visualizadas en el panel de variables.
- **Variables gráficas.** Un subconjunto de variables observadas que almacenan posiciones del array, y que se desea que sean señaladas en el visor del vector si se hace clic sobre ellas.
- **Variables visibles.** Un subconjunto de las variables gráficas que se desea que permanezcan señaladas durante toda la ejecución del algoritmo. Es preciso destacar que existen dos variables predefinidas que se pueden usar en el código fuente de forma global: "posicionA" y "posicionB". Se debe hacer uso de ellas si se quiere que estén señaladas siempre en el visor del vector, independientemente de si la ejecución se encuentra en el método principal o en métodos auxiliares.

Una vez cubierto el formulario se pasará a ejecutar ALT y, entre los algoritmos por defecto, aparecerá también el nuevo algoritmo del alumno.

4. Implementación

Esta sección describe brevemente algunos de los aspectos técnicos sobre la implementación llevada a cabo para ALT.

Tal y como se muestra en la Figura 9, los alumnos acceden a la herramienta a través de la página Web de la asignatura MTP, ya que ALT ha sido implementado como un Applet Java [6].

Una vez que se descarga e inicia dicho Applet, éste se conecta de nuevo al servidor Web para descargar los algoritmos, junto con un fichero de configuración que detalla cómo han de ser visualizados. Entre los algoritmos disponibles, se encontrarán tanto los que han sido creados por los profesores, como los que el alumno haya podido implementar durante su sesión de trabajo con la herramienta.

Una vez que el Applet, ejecutándose en la Máquina Virtual Java (JVM, *Java Virtual*

Machine) del navegador cliente, disponga del código fuente de los algoritmos, éstos serán compilados para luego pasar a ser ejecutados. Dicha ejecución es llevada a cabo por una JVM secundaria, controlada y monitorizada desde el Applet, que atenderá las peticiones del alumno (inicio, pausa, ejecución paso a paso, monitorización del estado del array y de las variables observadas, etc.).

Es preciso destacar que ALT ha sido implementado partiendo de otro proyecto anterior desarrollado en el Departamento de Informática, denominado *JavaTraceIt* [2,3]. Esta herramienta implementa un depurador y optimizador de aplicaciones Java y, entre otras funciones, permite la compilación de ficheros fuente y la ejecución paso a paso de los programas. Estas características están incluidas dentro de ALT y no han sido codificadas trabajando directamente con la librería Java destinada a tal fin (JPDA, *Java Platform Debugger Architecture*) [7], sino que se ha utilizado la API de JavaTraceIt, de más alto nivel y que ha facilitado en gran medida el trabajo realizado.

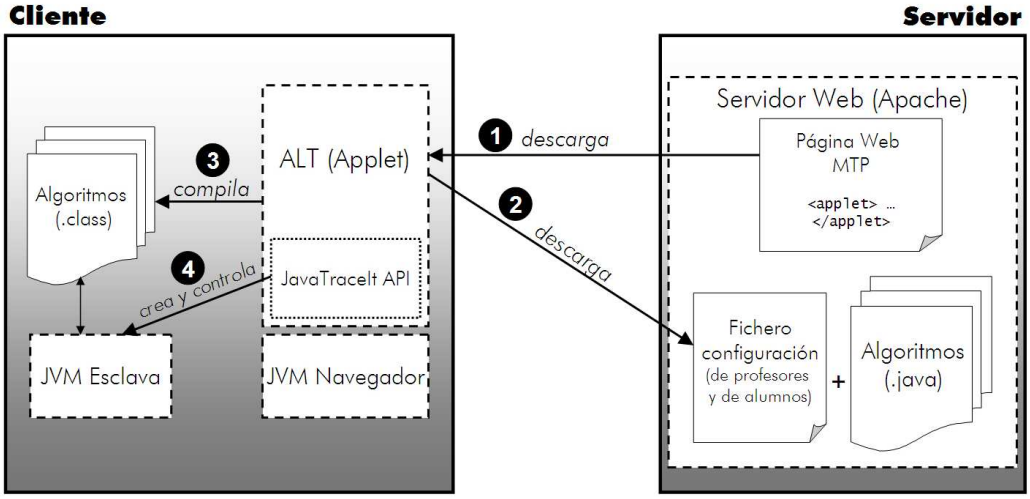

Figura 9. Arquitectura del sistema ALT

5. Conclusiones y Trabajo Futuro

Este artículo presenta una herramienta flexible y útil para la docencia de contenidos algorítmicos. Permite al alumno por un lado, ver los algoritmos de una manera gráfica y dinámica y, por otro, participar en la modificación de su código para ver cómo los cambios afectan a la "animación". Además está pensada para ser utilizada por el profesor en las clases presenciales como complemento a la pizarra.

ALT ha sido desarrollada recientemente y ya existe una versión totalmente funcional disponible en *http://trevinca.ei.uvigo.es/~rlaza/teoria.htm*. Se planea comenzar su utilización en el aula durante el próximo curso 2007-2008 en la asignatura MTP, por lo que todavía no se dispone de información acerca del nivel aceptación de la herramienta por parte de los alumnos. De todos modos, se confía en que será de gran ayuda porque ha surgido de una necesidad real detectada por el profesorado de la asignatura, que tiene varios años de experiencia impartiendo docencia de forma continuada.

El trabajo futuro se centrará principalmente en recoger las impresiones del alumnado durante el próximo curso, tratando de incorporar las aportaciones más interesantes. Sin embargo, se pueden anticipar varias mejoras interesantes como que el alumno pueda modificar el código fuente directamente sobre el visor y no tener que alterar el algoritmo mediante un formulario; la posibilidad de que se pudiesen visualizar no sólo vectores, sino árboles u otras estructuras; la capacidad para ocultar métodos auxiliares que sean superfluos para la animación o mejoras en la

interfaz, como por ejemplo el visor de pila que podría llegar a ser poco intuitivo para un alumno de primer curso de Informática.

Referencias

[1] Biliana, K.; Thiébaut, D. *Sorting Algorithms*. 1997. http://maven.smith.edu/~thiebaut/java/sort/demo.html.

[2] Glez-Peña, D.; Fdez-Riverola, F. *Understanding JPDA (debugging) & JVMTI (profiling) Java APIs within JavaTraceIt*. IADIS International Conference WWW/Internet 2006: ICWI, 2006.

[3] Glez-Peña, D.; Fdez-Riverola, F.; Méndez, J.R.; Díaz, F. *JavaTraceIt!: software didáctico de apoyo a la docencia en Java*. XI Jornadas de la Enseñanza Universitaria de la Informática: JENUI, 2005.

[4] Gosling, J.; Harrison, J.; Boritz, J. *Sorting Algorithms Demo*. 2001. http://www.cs.ubc.ca/~harrison/Java/sorting-demo.html

[5] Rosales, I. *Animador de Algoritmos*. Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla. http://www.lsi.us.es/docencia/asignaturas/ip1/trabajos/IndiceAnimador.htm.

[6] Sun Microsystems, Inc. *Applets*. Sun Developer Network. http://www.java.sun.com/applets.

[7] Sun Microsystems, Inc. *Java Platform Debugger Architecture (JPDA)*. 2002. http://java.sun.com/j2se/1.4.2/docs/guide/jpda/