Una herramienta educativa para mejorar la comprensión de algoritmos y estructuras de datos

Alejandra Schiavoni, Laura Fava, Jorge Rosso
LINTI - Laboratorio de Investigación en Nuevas Tecnologías Informáticas.
Facultad de Informática. Universidad Nacional de La Plata
Calle 50 esq. 120, 2do Piso. Tel: +54 221 4223528
{ales, lfava, jrosso}@info.unlp.edu.ar

Resumen

En Ciencias de la Computación la enseñanza de las estructuras de datos y los algoritmos asociados a ellas es de suma importancia ya que representan la base para el desarrollo de toda clase de aplicaciones. Los temas relacionados con la algorítmica y el uso de las estructuras de datos aparecen en asignaturas de los primeros años de las carreras dado que ellas son el fundamento de conceptos más avanzados. En general, se observa que a los estudiantes les resulta complejo comprender y asimilar estos temas totalmente novedosos para ellos.

Este artículo describe el diseño y arquitectura de un entorno educativo para facilitar el aprendizaje de estructuras de datos y algoritmos en el trayecto inicial de la carrera. La herramienta propuesta ofrece la posibilidad de contar con una representación visual de las estructuras de datos y de inspeccionar el estado de ejecución de algoritmos definidos por los alumnos. El objetivo de este entorno es permitir a los alumnos experimentar de manera de reducir la abstracción que conlleva el estudio de estos temas.

Palabras clave: algoritmos y estructuras de datos, visualización de estructuras de datos, recursión, programación, Programación Orientada a Aspectos, JAVA.

Introducción

En Ciencias de la Computación el estudio de las Estructuras de Datos, es fundamental dado que las mismas son utilizadas en el desarrollo de casi todo tipo de software. Estudiar estructuras de datos implica estudiar las diferentes maneras de almacenar y organizar datos para facilitar el acceso y modificación [1]. Por otro lado, la algorítmica es un campo primordial en el procesamiento de datos. Sin embargo, ciertos conceptos asociados a estos temas no siempre resultan simples de comprender para los alumnos. Knuth recomienda que la mejor forma de aprender y entender un algoritmo es probarlo mediante un ejemplo [2]. En muchos casos, aún conociendo el tema, resulta muy difícil comprender el funcionamiento de un algoritmo con sólo leerlo.

Los temas relacionados con la algorítmica y el uso de las estructuras de datos aparecen en asignaturas de los primeros años de las carreras de Ciencias de la Computación, ya que representan la base para el estudio de contenidos más avanzados. En particular, en la Facultad de Informática de la Universidad Nacional de La Plata, la materia *Algoritmos y Estructuras de Datos* es obligatoria y corresponde a segundo año de las carreras de grado: Licenciatura en Informática, Licenciatura en Sistemas y Analista Programador Universitario. Además, en segundo año de la carrera de Ingeniería en Computación se dicta la materia *Programación 3*, también de carácter obligatorio. Las asignaturas mencionadas contienen los temas relacionados con el uso y aplicación de

estructuras de datos básicas y avanzadas.

Este artículo describe el diseño y la arquitectura de un entorno educativo para facilitar el aprendizaje de algoritmos en el trayecto inicial de las carreras de ciencias de la computación. Este entorno podría ser usado como un complemento novedoso de las clases teóricas y prácticas en las que el alumno incorpora los conceptos y los aplica en el desarrollo de sus propios algoritmos. El uso de una herramienta de soporte para la enseñanza/aprendizaje que visualice las diferentes estructuras de datos y el comportamiento de los algoritmos sobre ellas resulta de suma importancia, especialmente para los estudiantes de los primeros años de las carreras de informática, que no están totalmente familiarizados con estos temas.

En las próximas secciones se expondrán las características más relevantes de algunas herramientas existentes con objetivos similares a la propuesta en este artículo. Luego, se describirá en forma general, la arquitectura y funcionalidad inicial del entorno propuesto, y las tecnologías a utilizar. El artículo finaliza con las conclusiones y trabajos futuros previstos para este proyecto.

Estado del arte: Análisis de herramientas existentes

Desde hace tiempo se vienen estudiando mecanismos complementarios al material teórico-práctico usado tradicionalmente para la enseñanza de las estructuras de datos y algoritmos. Se fueron desarrollando diversas herramientas basadas en simulación, animación y visualización que mejoran la comprensión y aprendizaje de los tópicos mencionados. Algunas consisten en animaciones de las operaciones sobre las estructuras a partir de un código fijo, que puede estar visible durante la ejecución del algoritmo, y puede estar totalmente inaccesible. Otras, si bien permiten visualizar la ejecución de un código propio, fuerzan a usar alguna técnica *intrusiva* -decorando o adaptando el código fuente original de la estructura- o *conductista* -usando un lenguaje no estándar/propietario-.

A continuación se describirán brevemente algunas herramientas que dan cuenta de los distintos enfoques mencionados, todas con el fin de mejorar el aprendizaje de las estructuras de datos.

La primera herramienta que se describe es ***VisuAlgo*** [3], un software interactivo y web para la visualización de algoritmos clásicos,. Es interactivo en el sentido que los estudiantes pueden ingresar datos y ejecutar los algoritmos predefinidos. La pantalla principal muestra un conjunto algoritmos que pueden ser ejecutados e interactuados. Durante la ejecución se despliega un panel que visualiza un script predefinido y fijo con la lógica del mismo. VisuAlgo permite visualizar el manejo de las estructuras básicas y algoritmos asociados a través de una interfaz de usuario gráfica, moderna y unificada, que incluye desde algoritmos básicos de manejo de listas hasta algoritmos complejos vinculados con grafos.

Otra de las herramientas analizadas, provee una completa visualización de las estructuras de datos más usadas, como arreglos, pilas, colas, árboles y grafos y la animación de las operaciones más comunes sobre ellas [4]. Esta herramienta también provee una animación de algoritmos simples definidos por el usuario, a través del uso de un lenguaje propio llamado ***JavaMy***. Para la escritura del código, cada usuario puede usar el editor de textos provisto o importarlo de un archivo. Las estructuras de datos disponibles a inspeccionar son las que provee el software como estructuras observables, que el usuario puede instanciar. El uso de esta herramienta le implica al usuario el esfuerzo de aprender un lenguaje específico, que si bien tiene una sintaxis similar a Java, presenta algunas diferencias en cuanto a la definición y organización de los archivos fuente.

La próxima herramienta que se describe se denomina ***CADILAG***, está siendo desarrollada

en la Universidade Estadual Paulista, en Presidente Prudente, Brasil [5]. Fue desarrollada en JavaFX, permitiendo que sea utilizada como aplicación de escritorio o basada en web. Ofrece un conjunto limitado de estructuras de datos con las cuales trabajar, y a partir de la selección de una de ellas, la herramienta permite visualizar el comportamiento de sus operaciones básicas. Contiene una ayuda general con explicaciones sobre las operaciones y una asistencia para entender la estructura estudiada al momento de visualizar la animación. Si bien, es posible interactuar de manera amigable con las estructuras de datos y comprender cómo funcionan las operaciones, no es posible que los estudiantes verifiquen el funcionamiento de su propio código.

Finalmente, analizamos ***jGRASP***, el último entorno de desarrollo implementado por el grupo de investigación GRASP (**G**raphical **R**epresentations of **A**lgorithms, **S**tructures, and **P**rocesses) de la Universidad Auburn en Alabama, USA [6]. Es un ambiente de desarrollo integrado, de libre acceso, que provee visualizaciones generadas automáticamente para mejorar la comprensión de las estructuras de control, diagramas de clases UML y del funcionamiento de las estructuras de datos. En relación al enfoque de nuestro análisis, esta herramienta cuenta con visualizadores dinámicos que intentan reconocer automáticamente las estructuras a partir del código Java y graficarlas en forma adecuada. Si bien la herramienta es muy interesante al momento de visualizar las estructuras, representa un ambiente de desarrollo específico para este fin, que no acompaña de una manera amigable la escritura del código de los algoritmos, dado que no brinda ayudas contextuales, respecto a paquetes, clases y métodos disponibles.

Cada herramienta descripta tiene su atractivo. En particular, la herramienta que se propone tiene un enfoque similar a jGRASP, puesto que trabaja sobre código Java elaborado por los estudiantes sin ningún tipo de restricción y propone una visualización automática a partir del mismo. En la siguiente sección se describe la arquitectura y funcionalidad de la herramienta propuesta.

Arquitectura de la herramienta propuesta

La herramienta propuesta está orientada a ofrecer una funcionalidad amplia y diversa basada en la posibilidad de contar con una representación visual de las estructuras de datos y sus algoritmos asociados. Asimismo, se consideraron aspectos para atacar los puntos críticos que se presentan al momento de aprender una nueva técnica de resolución de problemas. Tal es el caso de la recursión que involucra una forma distinta de pensar y encarar la solución a problemas [7, 8].

Por otro lado, dado que JAVA es el lenguaje utilizado para la implementación de las estructuras de datos en las asignaturas *Algoritmos y Estructuras de Datos* y *Programación 3*, se tuvieron en cuenta soluciones basadas en este lenguaje. En las actividades prácticas, los estudiantes utilizan Eclipse como ambiente de desarrollo, motivo por el cual se decidió extender este entorno en lugar de desarrollar una herramienta diferente. De esta manera, los estudiantes seguirán usando Eclipse, un entorno de amplia aceptación en la comunidad de desarrolladores JAVA, optimizado con la potencialidad del plug-in. Podrán ejecutar sus programas de la manera tradicional o usando la funcionalidades que provee el plug-in.

Las funciones ofrecidas por la herramienta serán:

- Visualización del estado de ejecución del código fuente JAVA implementados por los alumnos.
- Acceso al estado de las variables en cada uno de los llamados recursivos .
- Visualización de las estructuras de datos

asociadas al algoritmo que se está ejecutando.

La definición de la herramienta involucra principalmente la creación de un plug-in para la visualización de las estructuras de datos y la implementación de Aspectos para interceptar la ejecución del código.

1. *Desarrollo de un plug-in para Eclipse.*

Eclipse es un framework de código abierto para la creación de entornos de desarrollo integrados utilizando un conjunto de herramientas y componentes GUI pre-construidas. Desde su lanzamiento, ha generado gran interés en la comunidad de desarrolladores JAVA tanto por sus funcionalidades como por sus capacidades de extensión. El *workbench* de Eclipse se compone de editores, vistas y perspectivas.

Los *editores* son áreas de edición asociados con tipos de documentos específicos. Por ejemplo hay editores de archvos xml, .java, JSp, etc. brindando ayudas específicas para cada tipo archivo.

Las *vistas* son paneles que apoyan a los editores y en su conjunto, conforman una perspectiva.

Las *perspectivas* son combinaciones de vistas específicas y editores que dan soporte al usuario durante el desarrollo de una aplicación.

La implementación del plug-in[1] [9] propuesto proveerá a Eclipse de una nueva *perspectiva*, compuesta por diferentes *vistas*, para visualizar, durante la ejecución de algún algoritmo, el estado de la Recursión o de una Estructura de Datos particular. Se tiene previsto, en primera instancia, desarrollar una *perspectiva* con dos vistas, una para mostrar el estado de la Recursión y otra para la visualización de los Arboles Binarios y algoritmos de acceso. La Fig. 1 muestra de manera simplificada la arquitectura de Eclipse, donde se inserta el nuevo plug-in con dos vistas. Puede observarse que es posible agregar nuevas funcionalidades mediante la incorporación de nuevas vistas como Visualizador para Listas, Árboles Binarios de Búsqueda, Árboles Generales, Grafos, etc.

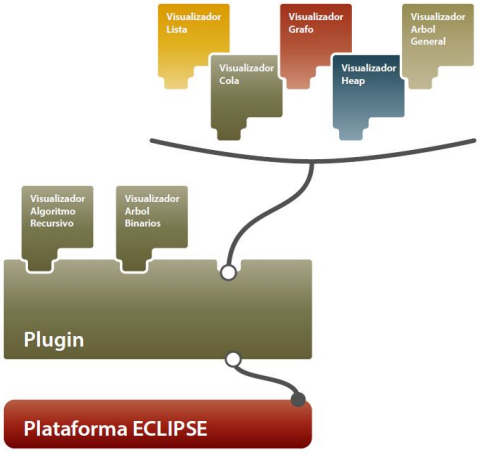

Figura 1: Arquitectura de Eclispe

2. *Definición de las representaciones visuales*

Como ya se describió en el punto anterior, el plug-in definirá una nueva *Perspectiva* con varias *Vistas* destinadas a visualizar las Estructuras de Datos y el estado de las llamadas en Algoritmos Recursivas. La plataforma estándar de JAVA, provee Swing, una librería de componentes gráficas para crear GUIs de alta calidad [10] que serán usadas para representar las Estructuras de Datos y animar las operaciones vinculadas con esas estructuras como por ejemplo: recorridos preOrden, postOrden, inOrden para los diferentes árboles o recorridos en amplitud o en profundidad para grafos. La Figura 2 muestra un boceto de la Perspectiva propuesta ejecutando un recorrido preOrden() sobre una estructura de árbol binario. La perspectiva está dividida en vistas, a la izquierda una mostrando los paquetes con sus

[1] Un plug-in es una componente de software, es la unidad funcional más pequeña de la plataforma Eclipse que puede ser desarrollada de manera separada e integrada al entorno Eclipse para potenciar su funcionalidad.

clases, en el centro otra con un editor mostrando el código fuente java y a la derecha una ventana de Visualización del Arbol Binario y los llamados recursivos.

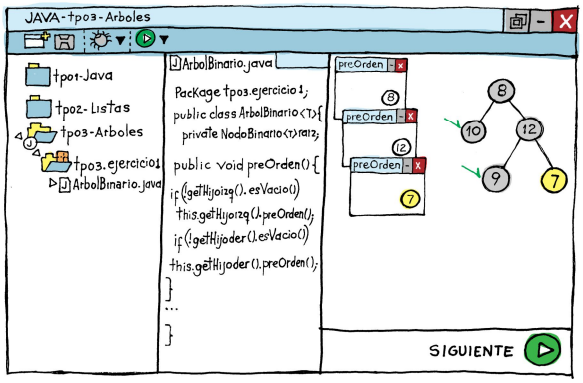

Figura 2: Nueva Perspectiva: Vista Recursión

Para visualizar la ejecución de los algoritmos se usarán diferentes mecanismos. En la imagen de la figura se muestran ventanas en cascada que corresponden a los llamados recursivos pendientes, nodos con tilde que indican que han sido visitados, un nodo amarillo indica que el algoritmo está parado en ese nodo, etc. Esta vista también tendrá mecanismos para conocer cómo manejar la estructura para poder dibujarla, ventanas para seleccionar que métodos serán interceptados para examinar los datos de los objetos.

3. *Definición de Aspectos y su aplicación en la herramienta*

Unos de los objetivos principales de la herramienta, fue poder visualizar las diferentes Estructuras de Datos, sin afectar o intervenir el código desarrollado por los estudiantes. Esto significa que la herramienta debe conocer las clases y métodos que permitirían describir la composición de cada estructura y los datos contenidos en ella, sin imponer restricciones. Es decir, el código de los estudiantes no tendrá que cumplir con convención de nombres, ni debería incluir anotaciones JAVA para indicar de qué tipo de estructura se trata y cuáles son sus métodos de acceso a la información. Por otro lado, utilizar archivos descriptores (con algún formato estándar como XML o JSON) asociados con las estructuras utilizadas, resultaría en una solución un tanto estática.

De esta manera llegamos a la conclusión de que la Programación Orientada a Aspectos nos permitiría resolver este tema de una manera más elegante y dinámica. Con AspectJ [11] es posible indicar puntos de un programa en los cuales uno desea examinar los objetos y realizar alguna determinada acción sin modificar el código original. La especificación de estos puntos de intervención (Pointcuts) y las acciones que se llevarán a cabo se escriben en un aspecto (Aspect).

El momento en que se entrelaza (weaving) el código origen con los aspectos se puede dar en tres momentos: en compilación (compile-time weaving), durante el empaquetado de un archivo JAR, o en ejecución (load-time weaving).

La Fig. 3 muestra una esquematización del funcionamiento de AOP para entremezclar los aspectos con los programas.

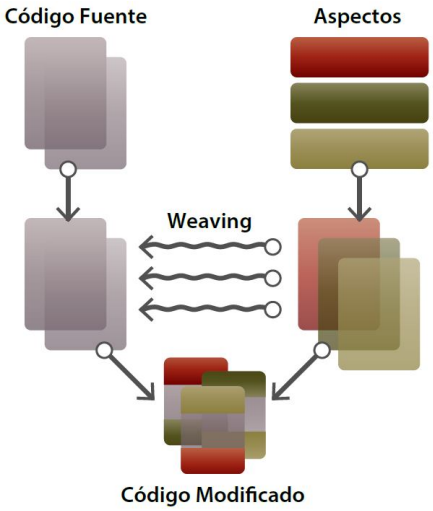

Figura 3: Weaving de Aspectos con el Código Fuente

Como puede observarse, los aspectos que definen qué código sería interceptado, son

independientes del código fuente implementado por los estudiantes. El uso de programación orientada a aspectos nos permite no ser intrusivo en la definición de las estructuras de datos y los algoritmos implementados por los estudiantes.

Conclusiones

En este artículo presentamos una herramienta destinada a mejorar el aprendizaje de algoritmos y estructuras de datos en los alumnos de los primeros años de las Carreras de Ciencias de la Computación. Esta herramienta de visualización no sólo permite a los estudiantes visualizar las estructuras de datos más comúnmente usadas, sino también le permite escribir en JAVA sus propios algoritmos y observar la ejecución dentro del mismo entorno de desarrollo ECLISPE. Lo destacable de la herramienta es que la animación de los algoritmos depende en forma directa del código propio que implementó y está ejecutando el alumno y se resuelve en tiempo de ejecución. El otro aspecto a resaltar, es la posibilidad que brinda la herramienta para realizar el seguimiento y análisis de los llamados recursivos, pudiendo inspeccionar el estado de las variables en cada llamado.

Atendiendo al uso permanente que hacen los jóvenes de las nuevas tecnologías y a la fuerte incorporación de las TICs en educación, creemos que esta herramienta va a ser un complemento efectivo a la enseñanza tradicional. A partir de los conocimientos adquiridos de las clases teóricas, la herramienta asistirá a los estudiantes en la creación de sus propios algoritmos, ayudando a disminuir la abstracción inherente al estudio de las Estructuras de Datos y algoritmos de acceso.

Como ya se mencionó en una primera versión, se implementará el Visualizador para Algoritmos Recursivos y el Visualizador para Arboles Binarios. Sin embargo la arquitectura es extensible, y podrían agregarse nuevos visualizadores para otros tipos de Estructura de Datos, sin afectar el código existente.

Con el primer prototipo del la herramienta se prevé el testeo con un grupo reducido y voluntario de estudiantes para evaluar el impacto en su aprendizaje. Se espera poder usarla como complemento innovador a las clases teóricas, explicaciones prácticas y textos recomendados por la Cátedra.

Referencias

[1] T. Cormen, C. Leiserson, R. Rivest, C. Stein; *Introduction to Algorithms*, Second Edition, ISBN 0-07-013151-1 (McGraw-Hill), 2001.

[2] D. Knuth, *The Art of Computing Programming*, Addison-Wesley, 1968

[3] S. Halim, Z. Koh, V. Loh, F. Halim, *Learning Algorithms with Unified and Interactive Web-Based Visualization*, Olympiads in Informatics, 2012, Vol. 6, 53–68, Vilnius University, 2012[4] T. Chen and T. Tarek Sobh; *A Tool for Data Structure Visualization and User-defined Algorithm Animation*, Department of Computer Science and engineering, University of Bridgeport, USA. Accesible en: http://www1bpt.bridgeport.edu/~risc/pdf/jp29.pdf

[5] G. Cardim, I. Marcal, C. de Sousa, D. de Campos, C. Marin; A. do Carmo, D. Toledo, A. Saito, R. Correia, R. Garcia, *Teaching and Learning of Data Structures Supported by Computer: An Experience with the CADILAG tool.* Information Systems and Technologies (CISTI), 2012 7th Iberian Conference, Madrid, p:1-5. ISBN 978-1-4673-2843-2

[6] J. Cross, D. Hendrix; *Workshop jGRASP: An Integrated Development Environment with Visualizations for Teaching Java in CS1, CS2, and Beyond*, 36th ASEE/IEEE Frontiers in Education Conference, ISBN 1-4244-0256-5, 27 -31 Oct. 2006.

[7] J. Zhang, M. Atay, E. Smith, E. Caldwell, E. Jones, *Using a Game-Like Module to Reinforce Student Understanding of Recursion*, Frontiers in Education Conference (FIE), IEEE, ISBN: 978-1-4799-3921-3, Madrid, Spain, 22 – 25 Oct, 2014.

[8] A. Chaffin, K. Doran, D. Hicks, T Barnes, *Experimental Evaluation of Teaching Recursion in a Video Game*, Proceedings of the 2009 ACM SIGGRAPH, Shimposium of Video Games, 79-86.

[9] A. Blewitt, *Eclipse 4 Plug-in Development by Example: Beginner's Guide.* ISBN 978-1-78216-032-8, Packt Publishing, 2013.

[10] C. Haase and R. Guy, *Filthy Rich Clients: Developing Animated and Graphical Effects for Desktop Java Applications*, ISBN-10: 0132413930, Addison Wesley, 2007.

[11] J. Gradecki, N. Lesiecki, *Mastering AspectJ: Aspect-Oriented Programming in Java.* ISDN 0-471-43104-4, Wiley, 2003.