ALGOLIPSE: una herramienta educativa para mejorar la comprensión de algoritmos y estructuras de datos

Laura Fava[1], Alejandra Schiavoni[1], Jorge Rosso[1], Facundo Falcone[1], Lisandro Ronconi[1],

[1] 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
{lfava, ales, jrosso}@info.unlp.edu.ar, faku_f@hotmail.com, lisandro.ronconi@gmail.com

Resumen. En Ciencias de la Computación la enseñanza de las estructuras de datos es de suma importancia puesto que representan la base para el desarrollo de toda clase de aplicaciones. El proceso de enseñanza-aprendizaje de estos conceptos acontece en los primeros años de las carreras, por lo que resultan mas complejos de aprehender debido, entre otras cosas, a la falta de abstracción observada en los estudiantes. En este contexto, surgieron numerosas herramientas de visualización de algoritmos, atractivas tanto para educadores, como para estudiantes, pero que no han sido ampliamente adoptadas debido al esfuerzo que implica su uso. En este trabajo, se presenta una extensión para Eclipse que genera automáticamnte visualizaciones de estructuras de datos y algoritmos a partir de código JAVA implementado por los estudiantes y no i ntervenido. Esta propuesta ayuda a reducir la abstracción que conlleva el estudio de estos temas sin el esfuerzo extra que implica aprender una nueva herramienta, lenguaje o librerías especiales para la visualización de algoritmos.

Palabras claves: visualización de estructuras de datos, visualización de algoritmos, recursión, programación orientada a aspectos (POA), JAVA.

1 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]. El propósito principal de la mayoría de los programas no es procesar información sino almacenarla y recuperarla, en lo posible de manera rápida. Las estructuras de datos, en términos generales, organizan o estructuran una colección de datos, requieren de un determinado espacio para almacenarlos y de una cantidad de tiempo para accederlos. En consecuencia, cuando se necesita resolver un problema, comúnmente se necesita seleccionar una determinada estructura de datos

considerando las restricciones de espacio y de tiempo. La introducción de estos temas complejos, a través de explicaciones teóricas y mediante su refuerzo con actividades prácticas, muchas veces no alcanza para que los estudiantes sean capaces de comprender el funcionamiento de todas las estructuras, poder determinar cuál es la mejor y codificar los algoritmos que representan una solución.

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, lo que aumenta la complejidad de su enseñanza-aprendizaje. En particular, en la Facultad de Informática de la Universidad Nacional de La Plata, la materia que comprende estos contenidos es Algoritmos y Estructuras de Datos, asignatura obligatoria que 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, que incluye los temas mencionados. En estas asignaturas se abordan los temas relacionados con el uso y aplicación de estructuras de datos básicas y avanzadas, algoritmos que permiten la manipulación de dichas estructuras y también el análisis de la eficiencia de los mismos.

Para que un curso sobre estructuras de datos alcance el éxito deseado debe contar con una componente fuerte de programación de las diferentes estructuras que se estudian [2], por ello, cada uno de los estructuras enseñadas tiene su parte práctica para reforzar la comprensión de las mismas. En muchos casos se observa que los estudiantes, aun conociendo las estructuras donde se guardan los datos y comprendiendo su organización, les resulta difícil entender el funcionamiento de un algoritmo con sólo leerlo. Donald Knuth, menciona que la mejor forma de aprender y entender lo que hace un algoritmo es probarlo mediante un ejemplo [3].

Con el fin de mejorar la comprensión de algoritmos, en los últimos años han aparecido numerosas herramientas de visualización y animación que permiten complementar el análisis, resultan muy útiles en procesos de auto-formación y favorecen la comprensión del problema en sí. Sin embargo, algunas se basan en contextos estáticos, no permitiéndole a los alumnos ejecutar su propio código, otras requieren esfuerzos extras para ser usadas, lo que desalienta su uso. En su mayoría, las propuestas resultan atractivas para los educadores quienes las consideran como ayudas pedagógicas, sin embargo no han logrado alcanzar popularidad para la enseñanza en ciencias de computación. Algunas razones podrían deberse a las escasas mejoras de aprendizaje observadas en comparación con el esfuerzo que implica usarlas. Muchas de ellas, como veremos más adelante en este artículo requieren que los estudiantes aprendan lenguajes o librerías específicas para escribir los algoritmos, otras incluyen códigos estáticos donde los estudiantes sólo pueden ver la animación de esos códigos y a menudo los aprendizajes son pasivos, es decir, el estudiante no puede interactuar con las estructuras y algoritmos.

Este artículo describe la arquitectura e implementación de una herramienta educativa para facilitar el aprendizaje de algoritmos de acceso a las estructuras de datos, que intenta paliar estas dificultades. 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. En las próximas secciones se describirán las características más relevantes de algunas herramientas existentes con objetivos similares a la propuesta en este artículo. Luego, se expondrá la arquitectura y funcionalidad del entorno propuesto, y por último, la implementación de la herramienta mencionando la metodología y las tecnologías que se utilizaron.

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

La primera herramienta que se describe es VisuAlgo [4], un software interactivo y web para la visualización de algoritmos clásicos. Es interactivo en el sentido que los estudiantes pueden ingresar datos y ejecutar ciertos algoritmos predefinidos. 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 herramienta analizada provee animación de algoritmos simples definidos por el usuario pero a través del uso de un lenguaje propio llamado JavaMy [5]. Para la escritura del código, cada usuario puede usar el editor de textos provisto o importarlo de un archivo. 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.

CADILAG es otra herramienta que está siendo desarrollada en la Universidade Estadual Paulista, en Presidente Prudente, Brasil [6]. 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. 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.

Otra herramienta analizada es jGRASP, el último entorno de desarrollo implementado por el grupo de investigación GRASP de la Universidad Auburn en Alabama, USA [7]. Es un ambiente de desarrollo integrado, de libre acceso, que provee visualizaciones generadas automáticamente para mejorar la comprensión 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.

Finalmente, sintetizamos las funcionalidades provistas por otra herramienta analizada denominada Algorithm Visualizer [8]. Esta herramienta es basada en web y provee un conjunto de algoritmos implementados sobre distintos temas: recorridos sobre árboles y sobre grafos, algoritmos de ordenación, algoritmos clásicos de programación dinámica y de backtracking, entre otros. Esta herramienta es muy completa y ofrece también la posibilidad de ingresar un código propio, pero usando un API propia llamanda Tracer API.

Cada herramienta descripta tiene su atractivo. En particular, la herramienta que se propone en este artículo 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, sin embargo, nuestra propuesta está basada en la extensión de las capacidades de Eclipse y no en la implementación de una herramienta independiente como es JGrasp. En las siguientes secciones se describen la arquitectura y las funcionalidades de la herramienta propuesta, y la implementación realizada.

3 Diseño de la herramienta propuesta

La herramienta propuesta está orientada a ofrecer una representación visual de la ejecución de algoritmos sobre estructuras de datos y la visualización de los llamados de algoritmos recursivos, que involucra una forma distinta de pensar y encarar la solución a problemas [9, 10]. 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, con el agregado de un nuevo plug-in denominado Algolipse.

La definición de la herramienta involucra principalmente la creación de un plug-in para la visualización de las estructuras de datos y de los llamdos recursivos y el uso de programción orientada a aspectos para interceptar la ejecución del código.

3.1 Arquitectura del plugin Algolipse en el entorno Eclispe

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 archivos 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] [11] propuesto proveerá a E clipse nuevas *Vistas*, dedicadas a visualizar la ejecución de algoritmos sobre una estructura de datos particular y en el caso de algoritmos recursivos, también se visualizará el estado de las variables en cada llamado recursivo.

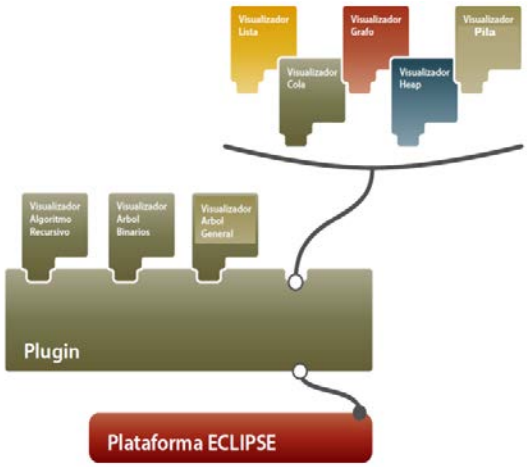

Fig. 1. Arquitectura de Eclipse

La Fig. 1 muestra de manera simplificada la arquitectura de Eclipse, donde se inserta el nuevo plug-in con varias visualizaciones. Puede observarse que es posible agregar nuevas funcionalidades mediante la incorporación de nuevas visualizaciones como Visualizador para Listas, Colas, Pilas, Grafos y otras estructuras.

[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.

3.2 Visualizadores de las Estructuras de Datos

Como ya se describió en el punto anterior, el plug-in definirá nuevas *Vistas* destinadas a visualizar las estructuras de datos y el estado de las llamadas de los algoritmos recursivas. Para interfaces gráficas la plataforma estándar de JAVA, provee librerías con componentes gráficas como Abstract Window Toolkit (AWT) y Swing [12], sin embrago, para el desarrollo de plugin, se recomienda el uso de una librería de componentes gráficas o widgets alternativa denominada Standard Widget Toolkit (SWT) [13]. Esta librería combina lo mejor de AWT y Swing, no es parte de la plataforma estándar JAVA pero se implementó para usar con ella y en especial para Eclipse. SWT fue desarrollada por IBM, y actualmente es mantenida por el Eclipse Fundación en conjunto con el IDE de Eclipse. Para mostrar elementos visuales, la implementación de SWT accede a librerías nativas de GUI del sistema operativo usando JNI (Java Native Interface) en una manera similar a AWT, es decir usando APIs específicas del sistema operativo, pero a diferencia de esta, cuenta con una rica librería de componentes como Swing. Esta librería está licenciada bajo la Eclipse Public License, una licencia open source aprobada por el Open Source Initiative.

La Fig. 2 muestra la arquitectura de la librería de componentes de interfaz de usuario estándar AWT/Swing y SWT. Las primeras se presentan juntas ya que Swing se construyó sobre AWT para proveer componentes más sofisticadas, flexibles y con un *look&feel* independiente de la plataforma, entre otras cosas. A la derecha se muestra las capas de SWT donde se observa como capa superior JFace, compuesto por un conjunto de herramientas de interfaz de usuario de alto nivel que utiliza los *widgets* de SWT para proporcionar componentes basados en modelos. JFace depende de SWT, pero no oculta los widgets SWT, dando la capacidad de elegir entre el desarrollo de interfaz de usuario basado en modelos o mediante manipulación de widgets en crudo.

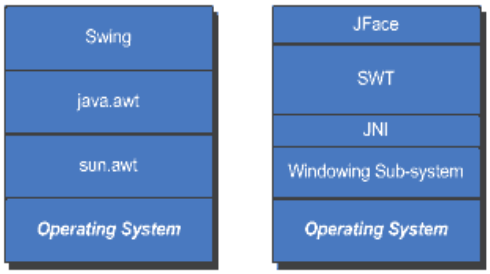

Fig. 2. Arquitecturas de AWT/Swing y SWT

En síntesis, SWT intenta hacer lo mejor de ambas librerías Swing y AWT, define una API portable y proporciona implementaciones para todas las plataformas. Siempre que sea posible, las implementaciones usan widgets nativos. Esto permite que se reflejen los aspectos de la interfaz de usuario gráfica del sistema operativo subyacente.

3.3 Programación Orientada a Aspectos en el plugin Algolipse

Uno 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 permitan 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 incluir anotaciones JAVA para indicar de qué tipo de estructura se trata y cuáles son sus métodos de acceso a l a 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 [14] 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 d el funcionamiento de AOP para entremezclar los aspectos con los programas.

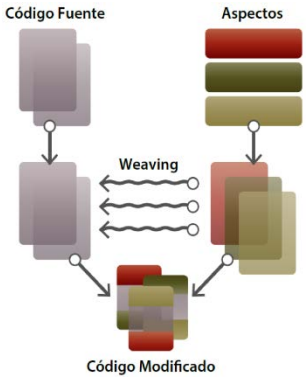

Fig. 3: Weaving de Aspectos con el Código Fuente

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 as pectos nos permite no ser intrusivo en la definición de las estructuras de datos y los algoritmos implementados por los estudiantes.

4 Implementación de la herramienta propuesta

Como se ha mencionado, muchas herramientas no han alcanzado demasiada popularidad debido a los beneficios obtenidos en relación al esfuerzo que implica su uso. Por este motivo, se tomaron dos decisions importantes de diseño tendientes a evitar la complejidad del uso: la primera está relacionada con mantener a l os estudiantes en el entorno de desarrollo habitual para probar sus códigos, por ello se ha implementado un plugin para ECLIPSE y la segunda determinación esta relacionada con no obligar a los estudiantes a aprender un nuevo lenguaje, pseudocódigo o API para obtener la visualización de sus producciones, por el contrario, pueden codificar normalmente sus algoritmos en JAVA con la API estándar.

4.1 El plugin Algolispe

El plugin Algolispe fue desarrollado siguiendo los lineamientos para desarrollo de cualquier plugin, su instalación es simple y automáticamente las nuevas Vistas estarán disponibles para usar en cualquiera de las Perspectivas de Eclipse.

Una vez instalado el plugin y agregada la nueva vista en alguna de las perspectivas, Eclipse mostrará como ilustra Figura 4. La nueva *Vista* de nombre *Algolispe* mostrará tres paneles internos: el visualizador de las estructuras de datos, selector de métodos de la clase seleccionada y que pueden ser inspeccionados y si el algoritmo es recursivo, aparecerán ventanas en cascada que representan cada uno de los llamados recursivos con el estado de las variables y la visualización de los algoritmos en ese llamado.

El código que se ejecuta, como ya mencionamos es JAVA. Los alumnos escriben sus clases como ListaEnlazada, ArbolBinario, ArbolAVL, ArbolGeneral, etc., sin ninguna restricción y pueden hacerlo además, con Tipos Genéricos, de manera que una implementación les sirva para instanciar estructuras que mantengan diferentes tipos de datos. Una vez que codifican las operaciones de sus estructuras, pueden visualizar su funcionamiento, indicando mediante la ventana de configuración, qué clase quieren visualizar y que algoritmo o algoritmos. Una vez elegida la clase, en el panel de la derecha aparecen los algoritmos que pueden visualizarse. En el caso de la Figura 4, se seleccionó la clase ArbolGeneral y el método printPreorden(). También podrían seleccionarse más métodos de la lista de algoritmos que pueden aplicarse a la estructura de datos ArbolGeneral.

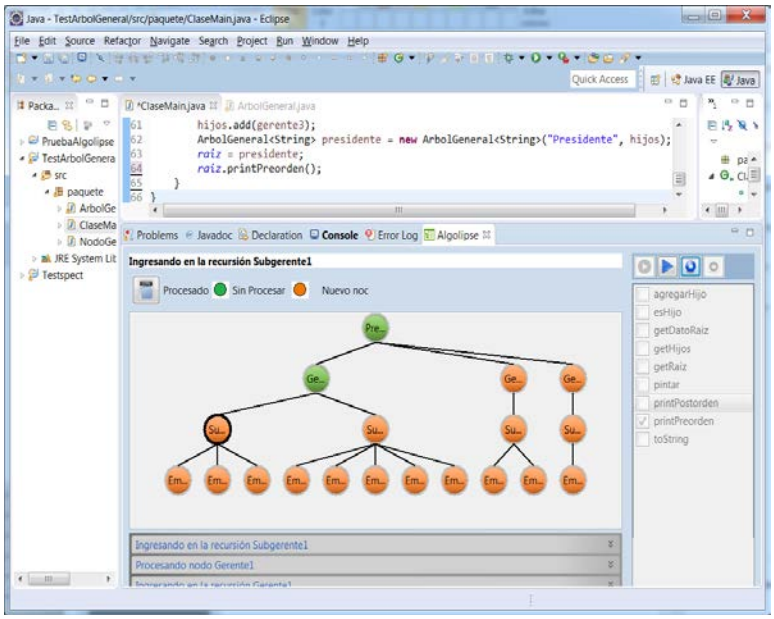

Fig. 4: Vistas del IDE Eclipse durante la ejecución del algoritmo printPreorden

Para ejecutar la aplicación, sólo es necesario crear una clase ejecutable, es decir, una clase que incluya el método main(String[] args). A continuación se muestra, a modo de ejemplo, segmentos código donde se crea el árbol de la Fig. 4 y finalmente se invoca al método printPreorden() que se quiere visualizar.

```
public class ClaseMain {
 public static ArbolGeneral<String> raiz;

 public static void main(String[] args) {
   LinkedList< ArbolGeneral<String> > hijos =
             new LinkedList< ArbolGeneral<String>>();
   hijos.add(new ArbolGeneral<String>("Empleado1"));
   hijos.add(new ArbolGeneral<String>("Empleado2"));
   hijos.add(new ArbolGeneral<String>("Empleado3"));
   ArbolGeneral<String> subGerente =
            new ArbolGeneral<String>("Subgerente1", hijos);
   . . .
   ArbolGeneral<String> presidente =
            new ArbolGeneral<String>("Presidente", hijos);
   raiz = presidente;
   raiz.printPreorden();
 }
}
```

4.2 El uso de AOP para visualizar código JAVA

El segundo aspecto que se tuvo en cuenta al momento de diseñar la herramienta está relacionado con el código JAVA escrito por los estudiantes. Durante la cursada de las asignaturas mencionadas, los alumnos desarrollan prácticas donde definen sus propias estructuras de datos, según cierta especificación UML, como Listas, Pilas, Colas, Arboles y Grafos y abundantes algoritmos asociados. Para permitir la visualización de los algoritmos a partir de código JAVA no modificado, utilizamos programación orientada a aspectos. El uso de aspectos permite generar visualizaciones automáticamente a partir de código JAVA sin necesidad de utilizar una API especial, metadatos como anotaciones o archivos de configuración XML. Los estudiantes codifican, compilan y ejecutan para visualizar.

La manera más apropiada que encontramos para interceptar la ejecución de un método, elegido en tiempo de ejecución, es decir durante la ejecución de códigos compilados, fue el uso de programación orientada a aspectos. De esta manera, cuando se alcanza una llamada a un método previamente seleccionado desde la interface de usuarios gráfica, el aspecto lo detectará para comenzar a actuar. Cabe destacar que para proveer mayor *feedback*, la visualización además muestra mediante diferentes colores que nodos han sido procesados y cuáles no. Para lograr esto también se ha utilizado aspectos.

5 Conclusiones y trabajos futuros

En este artículo presentamos una herramienta destinada a mejorar el aprendizaje de algoritmos y estructuras de datos, destinada a los alumnos de los primeros años de las Carreras de Ciencias de la Computación. Los estudiantes pueden escribir sus propios algoritmos en JAVA dentro de ECLIPSE como lo hacen habitualmente y observar la visualización de los mismos. Lo destacable de la herramienta es que la animación de los algoritmos depende en forma directa del código que está ejecutando el alumno. 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, consideramos 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.

En esta primera versión, se implementó el Visualizador para Algoritmos Recursivos y los Visualizadores para Listas, Pilas, Colas, Arboles Binarios y Arboles Generales. Actualmente se están comenzando a hacer las primeras pruebas de usabilidad de la herramienta para evaluar diferentes aspectos relacionados con la facilidad de uso, claridad

de la interfaz y eficacia en el funcionamiento. En una próxima etapa, se prevé realizar testeos con alumnos de las materias mencionadas a f in de evaluar el impacto en el aprendizaje.

Se espera poder incorporar Algolipse como complemento innovador a l as clases teóricas, explicaciones prácticas y textos recomendados por las cátedras. Asimismo esto permitirá retroalimentar la aplicación con sugerencias tanto por parte de los docentes como de los mismos estudiantes.

Referencias

1. T. Cormen, C. Leiserson, R. Rivest, C. Stein; Introduction to Algorithms, Second Edition, ISBN 0-07-013151-1 (McGraw-Hill), 2001.
2. C. Shaffer, Data Structures and Algorithm Analysis, Edition 3.2 (Java Version), Department of Computer Science, Virginia Tech, Blacksburg, VA 2406, January 2, 2012, Update 3.2.0.3.
3. D. Knuth, The Art of Computing Programming, Volume 1: Fundamental Algorithms, Third Edition, ISBN 0-201-89683-4 AddisonWesley LOngman, 1997.
4. 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.
5. 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
6. 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
7. 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.
8. Algorithm Visualizer. Disponible en GitHub: https://www.producthunt.com/tech/algorithm-visualizer
9. 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.
10. 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.
11. A. Blewitt, Eclipse 4 Plug-in Development by Example: Beginner's Guide. ISBN 978-1-78216-032-8, Packt Publishing, 2013.
12. C. Haase and R. Guy, Filthy Rich Clients: Developing Animated and Graphical Effects for Desktop Java Applications, ISBN-10: 0132413930, Addison Wesley, 2007.
13. M. Scarpino, S. Holder, St. Ng, L. Mihalkovic, SWT_JFace in action, ISBN: 1-932394-27-3, Manning Publications Co, 2009.
14. J. Gradecki, N. Lesiecki, Mastering AspectJ: Aspect-Oriented Programming in Java. ISDN 0-471-43104-4, Wiley, 2003.