Mejorando el aprendizaje de Procesadores de Lenguajes mediante visualizaciones

Francisco J. Almeida-Martínez, Jaime Urquiza-Fuentes, J. Ángel Velázquez-Iturbide
Universidad Rey Juan Carlos, Laboratorio de Tecnologías de la Información en Educación – LITE
{francisco.almeida,jaime.urquiza,angel.velazquez}@urjc.es

Resumen

Procesadores de Lenguajes es una asignatura bastante compleja para los estudiantes. Aunque las herramientas de generación automática facilitan su diseño, apenas existen herramientas educativas. En esta comunicación se describe la herramienta educativa basada en visualización VAST. Con las últimas mejoras desarrolladas, VAST es un entorno integrado de diseño de procesadores de lenguajes con facilidades de visualización. Tras utilizar la herramienta durante la parte del curso dedicada a análisis ascendente, se han detectado mejoras significativas en el aprendizaje de los estudiantes que usaron VAST. Además, VAST es reconocida por los estudiantes como una herramienta de alta calidad, fácil de usar y que ayuda a su aprendizaje.

1. Introducción

Asignaturas como *Procesadores de Lenguajes* o *Compiladores* son tradicionalmente costosas para los estudiantes. Aún así, la complejidad del diseño de procesadores de lenguajes[1] ha disminuido gracias a las herramientas de generación automática, como la pareja ampliamente conocida *Lex* y *Yacc* [11]. Éstas producen el código del analizador a partir de una especificación –léxica, sintáctica o de traducción–. Este tipo de herramientas se usan en la enseñanza, pero no están pensadas para su uso educativo, más bien se enseña a los alumnos a utilizar una herramienta del ámbito profesional. El hecho de conocer el manejo de una herramienta profesional es una ventaja, pero su aprendizaje suele ser bastante costoso. Existen multitud de herramientas de este tipo, pero difieren bastante en las especificaciones, tanto en la notación utilizada como en la organización de las especificaciones léxica y sintáctica.

Las tecnologías educativas basadas en visualizaciones tienen un claro potencial en este campo. Algunos de los conceptos fundamentales de estas asignaturas son el árbol sintáctico y su proceso de construcción. La visualización de estos conceptos es abordable, de hecho existen herramientas educativas que visualizan estos conceptos en el ámbito de la teoría de autómatas y lenguajes formales [17]. Sin embargo se echan en falta herramientas de este tipo en nuestro ámbito.

En esta comunicación se describe la última versión de VAST, una herramienta educativa basada en visualización que mejora el aprendizaje de los estudiantes en el diseño de procesadores, así como su evaluación educativa. El resto de la comunicación se estructura como sigue. En la sección 2 se revisa los trabajos relacionados. La sección 3 describe la herramienta con las última mejoras incorporadas. En la sección 4 se detalla la evaluación educativa. Finalmente, la sección 5 expone las conclusiones y líneas de trabajo futuro.

2. Trabajos relacionados

La aplicación de tecnologías de visualización a los procesadores no es nueva. Aparte de herramientas más relacionadas con los fun-

[1] nos referiremos a ellos como *procesadores* en el resto del artículo

damentos teóricos de la teoría de autómatas y lenguajes formales como JFlap [17] o SEFALAS[2], hemos encontrado dos clases de herramientas dependiendo de si permiten construir procesadores a los usuarios o no.

Las herramientas de la primera clase proporcionan visualizaciones del funcionamiento de procesadores pero no permiten generarlos, bien porque están desarrollados para reconocer un lenguaje concreto –ICOMP [4] y VisiCLANG [15]–, bien porque son meras herramientas de visualización –CUPV [9], APA[3] [10] y TREE-VIEWER [18]. Tan sólo BURGRAM [7] se podría considerar más cercano a la generación ya que admite especificaciones que usen la sintaxis de YACC, aunque la propia herramienta no genera el procesador.

Las herramientas de la segunda clase sí permiten la generación de procesadores, pero cada herramienta usa, de forma transparente al usuario, un generador distinto y a veces propio. Así, VCOCO [16] utiliza la herramienta de generación COCO/R [12]. Su visualización se basa en resaltar las líneas en ejecución correspondientes a la gramática, las especificaciones léxica y sintáctica, la cadena de entrada y el propio código del analizador. ANTLRWorks[4] es un entorno visual para el desarrollo de gramáticas con un claro enfoque profesional, para generar los procesadores usa la herramienta ANTLR[5]. Tanto VCOCO como ANTLRWorks trabajan con analizadores descendentes (LL). Se podrían alcanzar funcionalidades similares con las parejas CUPV [9]-CUP[6] o BURGRAM [7]-Yacc[11], pero el usuario necesitaría generar de forma manual y explícita el procesador, usando las herramientas de generación mencionadas. Las cuatro posibilidades mencionadas dependen de una notación propia, o de un sistema con el que están íntimamente ligadas. Este enfoque restringe su uso educativo ya que hace al generador y las visualizaciones totalmente dependientes. Así, el alumno debe esforzarse en aprender a utilizar diferentes herramientas: el formato de especificación, el proceso de construcción y la interpretación de mensajes de salida –informe sobre conflictos, matriz de transiciones o conjuntos de items–. Además, el profesor también tiene que familiarizarse con los distintos entornos, obligándole a utilizar un tiempo que probablemente creerá necesario para otras tareas educativas [14].

Nuestros esfuerzos se han dedicado a diseñar una herramienta de visualización que permita a los usuarios –profesores o estudiantes– realizar tareas como construir y visualizar sus propios procesadores, así como ser casi independiente[7] de las herramientas de generación aprovechando el esfuerzo de los usuarios en su aprendizaje.

3. Descripción de VAST

En esta sección se describe la herramienta VAST . Brevemente se mencionan sus características básicas, véase [3] para más detalles. Tras varias evaluaciones [1, 2] se decidió aplicar un conjunto de mejoras descritas al final de esta sección y cuya evaluación se detalla más adelante en esta comunicación.

3.1. Características básicas

El objetivo fundamental de VAST es el uso educativo eficaz de visualizaciones de procesadores de lenguajes. Por lo tanto debemos prestar tanta atención a los aspectos visualizados como al uso educativo [8] de la herramienta.

La visualización básica generada por VAST consiste en el árbol sintáctico construido durante el procesamiento de la cadena de entrada, es decir la aplicación de las distintas producciones gramaticales distinguiendo símbolos (nodos) terminales y no terminales. Para facilitar el manejo de árboles sintácticos de gran tamaño VAST proporciona dos vistas sincronizadas, global y detallada. El

[2] http://lsi.ugr.es/plweb/static/software.html, 2010

[3] Realmente esta herramienta no tiene nombre, hemos escogido el acrónimo del título del artículo donde se describe

[4] http://www.antlr.org/works/, 2010

[5] http://www.antlr.org/, 2010

[6] http://www2.cs.tum.edu/projects/cup/, 2010

[7] VAST está desarrollado en Java, y por lo tanto el generador deberá generar código Java

usuario se puede desplazar fácilmente sobre ellas así como variar el grado de detalle con el zoom gráfico y la expansión/resumen de subárboles. Además de estas características estáticas, VAST permite animar el proceso de construcción del árbol sintáctico. La reproducción de esta animación se realiza con controles típicos de vídeo. Para mejorar la comprensión del proceso de construcción del árbol sintáctico VAST mantiene de forma sincronizada visualizaciones de la pila y la cadena de entrada.

En cuanto al uso educativo, VAST se ha diseñado para permitir a los usuarios observar el funcionamiento de sus propios procesadores. Por un lado, la visualización permite facilitar la labor de diseño de procesadores, por otro se consigue integrar las visualizaciones de forma más natural en tareas y ejercicios, por ejemplo diseñar una gramática y demostrar su correcto funcionamiento construyendo una animación.

A pesar de las ventajas que la visualización pueda aportar, su uso se verá muy restringido si no se facilita su integración en los diferentes contextos educativos [13]. Por ello hemos diseñado VAST con enfoque genérico de forma que se pueda adaptar a: diferentes técnicas de procesamiento (LL, LR) y diferentes herramientas de generación automática de procesadores como Cup[8] o ANTLR[9]. Así, VAST se divide en un API de generación de la información a visualizar, llamada VASTapi, y la interfaz gráfica de visualización, llamada VASTview. Con esta estructura, el usuario tenía que anotar su especificación sintáctica con llamadas a VASTapi, construir el procesador con el generador que quiera usar, y al ejecutarlo obtendría la información visualizable mediante VASTview.

3.2. Mejoras realizadas

Tras evaluar la herramienta [1] se observó que el proceso para obtener las visualizaciones con VAST no se adaptaba a la manera real de construcción de compiladores, ya que añade dos etapas al proceso –anotación de la especificación y visualización–. La principal consecuencia era que los estudiantes tenían que trabajar con varias aplicaciones distintas al mismo tiempo –consola de comandos para generar, compilar y ejecutar, editor para la especificación sintáctica y la cadena de entrada, y VAST para visualizar el resultado– haciéndolo más proclive a errores de los estudiantes. Por ello se planeó una integración global compuesta de dos integraciones funcionales que adapta el proceso de visualización a la manera real de construcción de compiladores.

Se distinguen dos tareas durante el desarrollo de un analizador: la construcción y la ejecución. Así la integración global debe permitir, por un lado editar, anotar, generar y compilar una determinada especificación –la construcción del analizador visualizable con VAST. Y por otro, editar la cadena de entrada y seleccionar el analizador a ejecutar para crear la visualización –la ejecución del analizador y la visualización de su comportamiento. El resultado final facilitará que el ciclo de desarrollo de un procesador –especificación, generación y prueba(visualización)– se haga por completo con la misma herramienta, VAST.

3.2.1. Integración del proceso de edición, anotación, generación y compilación

Esta integración persigue aglutinar las fases previas a la compilación del analizador generado. En primer lugar, VAST va a permitir tanto editar especificaciones como importarlas. A continuación, dada una especificación determinada, la anotación se realizará de forma transparente al usuario y la generación y compilación del analizador se hará desde la misma interfaz.

Para hacer transparente el proceso de anotación hay que automatizarlo. Técnicamente esto es sencillo, pero hay que construir anotadores automáticos para cada generador de analizadores, perdiendo en cierto modo el enfoque genérico original. Desde el punto de vista del estudiante, hacer que la anotación sea transparente es más importante que perder generalidad. Desde el punto de vista del profesor, el enfoque genérico se sigue mantenien-

[8] http://www2.cs.tum.edu/projects/cup/, 2010
[9] http://www.antlr.org, 2010

do, porque si no hay anotador automático disponible siempre se podrá realizar la anotación de forma manual.

Finalmente, la generación del procesador y su compilación son simples llamadas a las aplicaciones correspondientes.

3.2.2. Integración del proceso de ejecución y visualización

Esta integración persigue aglutinar las tareas de ejecución del analizador y visualización de su comportamiento. El objetivo principal es permitir desde la propia interfaz de VASTview la edición de la cadena de entrada y la visualización del proceso.

La cadena de entrada se puede editar directamente o cargar desde un fichero. Finalmente, para conseguir agrupar todos los casos de uso posibles también se contempla la posibilidad de cambiar el analizador utilizado, de forma que el usuario podría ejecutar un analizador previamente construido sin abandonar la interfaz de VASTview. La usabilidad de esta integración se ha evaluado de forma satisfactoria previamente [2].

3.2.3. Otras funcionalidades añadidas

Además de las integraciones antes descritas se ha mejorado tanto la aplicación como la interfaz de VASTview. Las mejoras en la aplicación han proporcionado mejoras en los tiempos de ejecución, acceso a un menú de ayuda y facilidad de configuración como la autodetección de la máquina virtual de Java.

Se ha mejorado VASTview de dos formas. Por un lado se ha flexibilizado la distribución general de la interfaz, de forma que se puede adaptar más fácilmente a la visualización de árboles de gran anchura o profundidad. Por otro, se han aumentado las vistas disponibles con la gramática asociada a la especificación del procesador, explicaciones textuales de las acciones realizadas durante el procesamiento de la cadena de entrada y la visualización de árboles sintácticos parcialmente construidos consecuencia de cadenas sintácticamente erróneas. El aspecto final de la interfaz de VASTview se puede ver en la Figura 1.

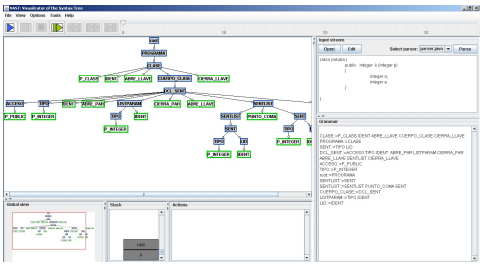

Figura 1: Interfaz mejorada de VASTview

4. Descripción de la evaluación

En esta sección se describe detalladamente la evaluación: los estudiantes participantes, el diseño experimental, el protocolo y las tareas educativas.

4.1. Participantes

En esta evaluación participaron de forma voluntaria 16 alumnos de la asignatura Compiladores e Intérpretes de la Universidad Rey Juan Carlos. La distribución por géneros no está equilibrada, pues sólo el 6.3 % (1/16) de los estudiantes eran mujeres. Antes de comenzar con la evaluación, se realizó un test de estilos de aprendizajes [6], sus resultados muestran que los estudiantes son principalmente visuales y activos.

4.2. Diseño experimental

Esta evaluación se ha diseñado como un estudio de eficacia educativa y usabilidad. Se dividió a los alumnos en dos grupos de tal forma que, el grupo de tratamiento utilizaría VAST y el grupo de control las herramientas JFlex-Cup para solucionar los ejercicios. Para equilibrar en la medida de lo posible las condiciones del experimento un mismo profesor impartió la parte teórica en ambos grupos y otro, externo al equipo de investigación, impartió la parte práctica también en ambos grupos.

Por lo tanto, la variable independiente es la herramienta utilizada: VAST para el grupo de tratamiento y JFlex-Cup para el grupo de control. Las variables dependientes se centran

en la eficacia pedagógica y la usabilidad. La eficacia pedagógica se ha medido con un pre-post-test diseñado según 4 niveles de la taxonomía de Bloom [5] –conocimiento, comprensión, aplicación y síntesis–, valorando los niveles de forma independiente y conjunta –media ponderada de los niveles con pesos del 1 al 4 respectivamente–. La usabilidad se midió en términos de la opinión de los estudiantes sobre cuatro aspectos fundamentales: facilidad de uso, mejora del aprendizaje, calidad de la herramienta y satisfacción del estudiante.

A diferencia de evaluaciones previas [1, 2] en las que la duración de las mismas era de una sesión de prácticas (2 horas), ésta se ha diseñado como una evaluación de larga duración en la que se han dedicado 4 sesiones (8 horas).

4.3. Tareas

Las tareas realizadas por los estudiantes debían ser documentadas al final de la evaluación, mediante respuestas textuales y visualizaciones (sólo en el grupo de tratamiento). Las tareas consistían en tres ejercicios sobre el diseño de gramáticas para analizadores LR(1). Antes de comenzar la evaluación se explicó al grupo de tratamiento cómo trabajar con VAST y con su editor. En el grupo de control se explicó cómo trabajar con las herramientas JFlex-Cup en conjunto, así como la generación y ejecución de los analizadores creados.

El *grupo de tratamiento* debía solucionar los ejercicios utilizando el editor de gramáticas de VAST y probando las diferentes cadenas de entradas proporcionadas. Las soluciones a cada ejercicio consistía en la especificación JFlex y Cup del analizador, junto con las cadenas de entradas probadas, acompañadas de las visualizaciones y explicaciones textuales correspondientes.

El *grupo de control* debía trabajar únicamente con las herramientas JFlex y Cup. Al igual que en el grupo de tratamiento, se les proporcionaba ciertas cadenas de entrada que debían probar. Los alumnos debían generar una gramática que aceptara dichas cadenas pero sin generar visualizaciones. En este caso, las soluciones a los ejercicios consistían únicamente en las especificaciones de JFlex y Cup y las explicaciones textuales correspondientes.

Grupo tratamiento	**Grupo Control**
Teoría (1)	
Estilos de aprendizaje(1)	
Pre-test (1)	
Eval. VAST(2)	Eval. JFlex-Cup(2)
Post-test(2)	
C. opinión (2)	
	C. de ayuda (1)
	VAST(1) C. opinión(1)

Cuadro 1: Protocolo de la evaluación. Los números (1 ó 2) hacen referencia al profesor que impartió esa sesión

Nivel	**Ctrl.**	**Trat.**	**Estadísticas**
K	0,36	0,45	t(13)=3.84 (p>0.05)
U	0,72	0,37	t(13)=0.29 (p>0.05)
Ap	0,37	0,10	t(13)=5.75 (p>0.05)
S	0,25	0,06	t(13)=5.00 (p>0.05)
Total	2,74	1,40	t(13)=6.25 (p>0.05)

Cuadro 2: Comparativa de los resultados del pre-test entre los grupos de control (Ctrl.) y tratamiento (Trat.). K (Conocimiento), U (Comprensión), Ap (Aplicación) y S (Síntesis)

4.4. Protocolo

En esta sección se describe el protocolo seguido para la realización de la evaluación. En el cuadro 1 se muestra un resumen de las actividades realizadas por los alumnos y profesores.

La parte teórica del análisis sintáctico duró un total de 10 clases (20 horas) en cada grupo. Dos semanas antes de la evaluación se realizó el test de estilos de aprendizajes [6]. Una semana antes de la evaluación se realizó el pre-test de conocimientos en ambos grupos. Este test reveló (ver cuadro 2) que no existían diferencias significativas entre los grupos. En la primera sesión del grupo de tratamiento se dedicó 30 minutos para introducir VAST. Durante el resto de la sesión los alumnos trabajaron en el primer ejercicio de la evaluación.

Al concluir la evaluación se realizó el post-test y se contestó un cuestionario de opinión

acerca de las herramienta utilizada. Después de la sesión de evaluación, se pidió a los alumnos del grupo control que contestaran a un cuestionario donde se les preguntaba si les hubiera resultado de ayuda alguna herramienta informática y una breve descripción de la misma. En la siguiente sesión (únicamente para el grupo de control) se introdujo VAST y se pidió que resolvieran de nuevo el primer ejercicio de la evaluación. Tras la familiarización con VAST, se pidió que contestaran al cuestionario de opinión acerca de VAST.

4.5. Resultados

En esta sección se detallan los resultados obtenidos durante el periodo de evaluación. Durante este tiempo se observó como los estudiantes trabajaban con la herramienta, así como los problemas que tuvieron en su manejo. Los resultados se dividen en resultados educativos, observaciones del instructor y las respuestas a los cuestionarios.

Tras finalizar la evaluación se analizó en primer lugar el tiempo de estudio de los alumnos involucrados en ambos grupos. No se obtuvo diferencia significativa que indicara que un grupo se hubiera implicado más con la asignatura después de realizar la evaluación ($t(9)=3{,}15,\ p>0.05$).

En el cuadro 3 se muestran las diferencias entre post-test y pre-test. Como se puede observar existen diferencias significativas en los niveles de comprensión y síntesis, es decir, los alumnos que utilizaron VAST, han incrementado significativamente sus niveles de comprensión y síntesis.

Durante las sesiones de la evaluación el instructor observó cómo los estudiantes trabajaban con la herramienta. En general, ninguno se vio en la necesidad de utilizar papel para diseñar las gramáticas que se pedían en el enunciado.

En el grupo de tratamiento se detectó la necesidad de que VAST mostrara el árbol sintáctico aunque la cadena de entrada no se adaptara a la gramática, de esta forma *sería más sencillo arreglar la gramática*. En el grupo de control los alumnos tuvieron problemas para generar el analizador y compilar las clases generadas en Java.

Nivel	**Ctrl.**	**Trat.**	**Estadísticas**
K	0,23	0,00	t(9)=0.28 (p>0.05)
U	-0,20	0,11	t(9)=0.95 (**p<0.05**)
Ap	0,03	0,02	t(9)=0.79 (p>0.05)
S	0,25	0,75	t(9)=0.21 (**p<0.05**)
Total	0,11	0,35	t(9)=0,69 (**p<0.05**)

Cuadro 3: Comparativa de los resultados de diferencias entre post-test y pre-test. K (Conocimiento), U (Comprensión), Ap (Aplicación) y S (Síntesis)

Aspecto	**Media**	**% [4-5]**
Facilidad de uso		
Facilidad de uso general	4,67	100 %
Promedio partes	4,10	73 %
Ayuda a la comprensión		
Construcción del AS	4,17	83 %
Funcionamiento de la pila	4,00	67 %
Procesamiento de la CE	3,67	50 %
Calidad técnica		
Calidad general	4,00	83 %
Promedio partes	3,35	62 %
Satisfacción de los estudiantes		
Satisfacción general	4,17	83 %

Cuadro 4: Puntuaciones de aspectos generales de VAST

El cuestionario de opinión de VAST se diseñó para que los alumnos pudieran tanto puntuar la herramienta como dar su opinión mediante respuestas abiertas. Aunque el número de alumnos era reducido se ha realizado un análisis de las puntuaciones medias obtenidas en los cuestionarios, los cuales permiten obtener información acerca de la facilidad de uso y calidad general y/o por partes y la ayuda a la compresión de diferentes partes del procesamiento.

En el cuadro 4 se muestran las puntuaciones medias obtenidas para los aspectos generales del grupo de tratamiento. Este grupo piensa que VAST es fácil de usar (4,67). Además, opinan que les ayuda a la comprensión de construcción del árbol sintáctico (4,17) y el funcionamiento de la pila (4,00). Por último,

Aspectos	Media	%[4-5]
Facilidad de uso		
Facilidad de uso Cup	4,5	100 %
Facilidad de uso JFlex	3,5	66,67 %
Promedio partes	4,11	76 %
Ayuda a la comprensión		
Construcción del AS	4,2	80 %
Funcionamiento de la pila	3,6	60 %
Procesamiento de la CE	4,4	100 %
Calidad técnica		
Calidad general	4,4	100 %
Promedio partes	4,14	72,67 %

Cuadro 5: Puntuaciones de aspectos generales de JFlex-Cup

opinan que la calidad general de la herramienta es alta (4,00).

En el cuadro 5 se muestran las puntuaciones medias obtenidas para los aspectos generales del grupo de control. En este grupo los alumnos piensan que las herramientas JFlex-Cup son fáciles de usar (4,00). Asimismo, opinan que les ayuda a comprender mejor la construcción del árbol sintáctico (4,2) y procesamiento de la cadena de entrada (4,4). Por último, la calidad general de las dos herramientas obtiene una alta puntuación (4,4).

En cuanto a la posibilidad de tener una herramienta informática para el desarrollo de las prácticas anteriores, el 85,71 % de los alumnos opinó que sí les hubiera servido de ayuda. Y sobre sus caraterísticas, 3 alumnos destacaron la posibilidad de crear un IDE para escribir las especificaciones (léxica y sintáctica). Por otro lado, 2 alumnos se centraron en facilitar la compilación. Por último, 2 alumnos se centraron en aspectos de visualización, creyendo necesario visualizar el árbol sintáctico y el autómata generado.

5. Conclusiones y trabajos futuros

A partir de un uso continuado de VAST se han obtenido mejoras significativas en el aprendizaje de los estudiantes. Además, opinan que es un entorno de calidad y fácil de usar.

Tras la realización de la evaluación se han obtenido diferencias significativas en dos de los niveles de la taxonomía de Bloom (comprensión y síntesis) y en el aprendizaje global de los alumnos. Las integraciones realizadas, el planteamiento de los ejercicios de la evaluación y la duración de la misma han permitido que los alumnos que utilizaban VAST se familiarizaran con la herramienta. Se ha comprobado que los alumnos que utilizaron VAST mejoraron un 31 % en el nivel de comprensión y un 50 % en el de síntesis. Además, aquellos que utilizaron la herramienta, mejoraron el aprendizaje global en un 24 % (véase el cuadro 3).

En cuanto a las respuestas a los cuestionarios de opinión el grupo de tratamiento piensa que VAST es fácil de usar y que les ayuda a la comprensión del procesamiento de la cadena de entrada, construcción del árbol sintáctico y pila. Al grupo de control, le llevó más tiempo adaptarse con las herramientas JFlex-Cup aunque las puntuaciones de los diferentes aspectos, revelan que los alumnos opinan que es fácil de usar. El mayor problema del grupo de control, se centra en la falta de un entorno que les permita editar gramáticas y generar analizadores sin necesidad de trabajar con la línea de comandos y directamente con Java.

El trabajo con VAST no ha terminado. A partir de estos resultados planteamos la incorporación de nuevas funcionalidades en el editor de gramáticas como destacar palabras clave y facilitar la edición de la cadena de entrada desde el propio editor. Además, planeamos una evaluación de larga duración con un grupo más representativo.

Agradecimientos

Los autores agradecen a la profesora Diana Pérez Marín su colaboración al impartir las clases prácticas descritas en esta comunicación. Este trabajo se ha financiado con el proyecto TIN2008-4103 del Ministerio de Ciencia y Tecnología del Reino de España.

Referencias

[1] F. Almeida-Martínez and J. Urquiza-Fuentes. Syntax trees visualization in language processing courses. In *Proc. of the 9th IEEE International Conference on*

Advanced Learning Technologies, ICALT 2009, pages 597–601, Los Alamitos, USA, 2009. IEEE Computer Society Press.

[2] F. Almeida-Martínez, J. Urquiza-Fuentes, and J. Velázquez-Iturbide. Softwareeducativo en procesadores de lenguajes: del enfoque genérico al enfoque centrado en el estudiante. In *Actas del IX Simposio Internacional de Informática Educativa, SIIE 2009*. Universidad de Coimbra, 2009.

[3] F. Almeida-Martínez, J. Urquiza-Fuentes, and J. Velázquez-Iturbide. Visualization of syntax trees for language processing courses. *Journal of the Universal Computer Science*, 15(7):1546–1561, 2009.

[4] K. Andrews, R. R. Henry, and W. K. Yamamoto. Design and implementation of the uw illustrated compiler. *SIGPLAN Not.*, 23(7):105–114, 1988.

[5] B. Bloom, E. Furst, W. Hill, and D. Krathwohl. *Taxonomy of Educational Objetives: Handbook I, The Cognitive Domain*. Addison-Wesley, 1959.

[6] R. Felder and L. Silverman. Learning and teaching styles in engineering education. *Engr. Education*, 78(7):674–681, 1988.

[7] C. García-Osorio, C. Gómez-Palacios, and N. García-Pedrajas. A tool for teaching ll and lr parsing algorithms. In *ITiCSE '08: Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 317–317, New York, NY, USA, 2008. ACM Press.

[8] C. Hundhausen, S. Douglas, and J. Stasko. A meta-study of algorithm visualization effectiveness. *J. of Vis. Lang. and Comp.*, 13(3):259–290, 2002.

[9] A. Kaplan and D. Shoup. CUPV a visualization tool for generated parsers. *SIGCSE Bull.*, 32(1):11–15, 2000.

[10] S. Khuri and Y. Sugono. Animating parsing algorithms. *SIGCSE Bull.*, 30(1):232–236, 1998.

[11] J. R. Levine, T. Mason, and D. Brown. *Lex & Yacc*. O'Reilly, second edition, 1992.

[12] H. Mössenböck. A generator for production quality compilers. In *CC '90: Proceedings of the third international workshop on Compiler compilers*, pages 42–55, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[13] T.Ñaps, S. Cooper, B. Koldehofe, C. Leska, G. Rößling, W. Dann, A. Korhonen, L. Malmi, J. Rantakokko, R. J. Ross, J. Anderson, R. Fleischer, M. Kuittinen, and M. McNally. Evaluating the educational impact of visualization. In *ITiCSE-WGR '03: Working group reports from ITiCSE on Innovation and technology in computer science education*, pages 124–136, New York, NY, USA, 2003. ACM Press.

[14] T.Ñaps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. Velázquez-Iturbide. Iticse 2002 working group report: Exploring the role of visualization and engagement in computer science education. *SIGCSE Bull.*, 35(2):131–152, june 2002.

[15] D. Resler. Visiclang—a visible compiler for clang. *SIGPLAN Not.*, 25(8):120–123, 1990.

[16] R. D. Resler and D. M. Deaver. VCOCO: a visualisation tool for teaching compilers. *SIGCSE Bull.*, 30(3):199–202, 1998.

[17] S. Rodger. Learning automata and formal languages interactively with JFLAP. *SIGCSE Bull.*, 38(3):360–360, 2006.

[18] S. R. Vegdahl. Using visualization tools to teach compiler design. *J. Comput. Small Coll.*, 16(2):72–83, 2001.