Ambiente web para la construcción, simulación y visualización 3D de autómatas finitos y lenguajes regul

Claudia Noguerón, Manuel Chim, Juan Carlos Tovilla, Leandro Balladares

Centro de Investigación en Computación del Instituto Politécnico Nacional,
Unidad Profesional "Adolfo López Mateos", Zacatenco, C.P. 07738, México, México
cnogo@hotmail.com, manuelchi@hotmail.com,
jsarmiento@sagitario.cic.ipn.mx, ballad@cic.ipn.mx

Resumen. En las ciencias computacionales se usan frecuentemente modelos abstractos. El aprendizaje de estos modelos es difícil y muchas veces requiere de un esfuerzo mental mayor que en otras áreas del conocimiento por parte los estudiantes cuando se usan los métodos de aprendizaje tradicionales. Lo anterior, demanda la búsqueda de métodos que asistan al estudiante durante el aprendizaje de dichos modelos. Hemos desarrollado un software de aprendizaje basado en el Web que busca ayudar al estudiante a comprender mejor los principios de construcción de compiladores, en particular la etapa de análisis léxico. El software ofrece un ambiente en el que son presentados de manera gráfica y animada las definiciones y algoritmos más importantes. Las animaciones presentadas, muestran como se construyen los autómatas finitos a partir de expresiones regulares, así como la operación del autómata en tres dimensiones (3D). Se presentan los principios de diseño e implementación del software desarrollado, así como el trabajo que aún se está desarrollando (trabajo futuro). Finalmente, se describen algunos trabajos relacionados.

1 Introducción

La tarea diaria de un profesor / difundidor de las ciencias de la computación señar conocimiento abstracto y promover el entendimiento correcto de este miento entre sus estudiantes. Por ejemplo, cuando un profesor quiere descri operación de los autómatas de pila o de los analizadores sintácticos ascendent lo único que tiene a la mano en el salón de clases es un pizarrón y tal vez plumones de distintos colores. El reto para el profesor, es explicar el funcion de dichos modelos mediante un ejemplo, haciendo uso únicamente de los mate que tiene al alcance y de su habilidad para explicar este tipo de conceptos. de tres o cuatro pasos el profesor comienza a borrar símbolos (en el caso del zador LR) y estados para agregar otros (en el caso del autómata de pila) de Los estudiantes, tienen que esforzarse más por reconstruir la secuencia de ope- nes complicadas de escritura y borrado y descubrir un sentido en el desorden, entender la funcionalidad de los modelos que el profesor intenta explicar. La

A. Gelbukh, M. Hernández Cruz (Eds.) Avances en la ciencia de la computación en Mé
CORE-2003, México.

lización y simulación gráfica de los autómatas de pila y de los analizadores sintácticos es una posible solución a este dilema.

En este trabajo, se describe un software de aprendizaje basado en el Web, desarrollado en el Centro de Investigación de Computación del IPN por alumnos del curso de Compiladores, que busca ayudar a los estudiantes a comprender mejor los principios de construcción de compiladores, en particular la etapa de análisis léxico. El software ofrece un ambiente en el que son presentados de manera gráfica las definiciones y algoritmos más importantes. Las animaciones presentadas, muestran gráficamente como se construyen los autómatas finitos a partir de expresiones regulares, así como la operación del autómata. Este software, formará parte de un libro electrónico basado en el Web del curso de Autómatas y Compiladores que actualmente está siendo desarrollado en el centro y que pretende integrar animaciones gráficas 3D de los algoritmos de cada una de las fases de un compilador.

2 Animación del algoritmos

La animación de algoritmos se refiere a ilustrar el comportamiento de un programa visualizando sus operaciones fundamentales mientras está en ejecución. Se ha probado, que estas visualizaciones son de utilidad en los campos de investigación y de educación en el diseño y análisis de algoritmos [5], [27], [28], [29], [34], [35]. Los primeros sistemas de animación de algoritmos eran dependientes de la plataforma [2–4], [21]. Con el advenimiento de las nuevas tecnologías, la popularidad del Web y la evolución del lenguaje de programación Java, los desarrolladores se han inclinado por construir sistemas de animación de algoritmos en línea, siendo de gran utilidad para educación a distancia [6], [7], [10–12], [16], [20], [26], [32], [33].

Los sistemas de animación de algoritmos han sido extensamente utilizados como soporte de enseñanza en las ciencias de la computación. Se han llevado a cabo diversos estudios para evaluar su efectividad en la educación, pero los resultados varían. Por ejemplo Brown [4], usó un sistema llamado BALSA-I para enseñar en cursos de Introducción a la Programación y Estructuras de Datos. El sistema fue utilizado como un visualizador de programas en el primer curso y como animador de algoritmos de alto nivel en el segundo. Brown, reportó que el uso de animaciones como complemento en algunos temas, incremento la rapidez de comprensión del conocimiento por parte de los alumnos en comparación con los temas en dónde no eran usadas animaciones. Otros estudios que han reportado resultados positivos en la utilidad de la animación de algoritmos para enseñanza son [13], [15], [24], [26], [32], [33]. Algunos de estos reportes indican que las animaciones deben de presentarse a los alumnos como material complementario y que debe de haber una explicación del funcionamiento de las animaciones por parte del instructor antes de que los estudiantes las utilicen, lo anterior con la finalidad de facilitar el aprendizaje. Otros trabajos, reportan que la animación de algoritmos debe de utilizarse únicamente como soporte para realización de tareas por parte de los estudiantes. Los trabajos anteriores, concluyen y confirman la utilidad pedagógica de enseñar con

animación de algoritmos en lugar de utilizar los métodos tradicionales (texto, genes).

Algunos autores [8], [9], [13–15], [19], [28], coinciden en que para que los sistemas de animación de algoritmos sean efectivos y benéficos para los usuarios, debe cuidarse su diseño y la utilización de las animaciones, además de respetar los siguientes lineamientos:

- *Acceso abierto.* Los usuarios deben tener acceso abierto a las animaciones, es decir, además de tener acceso al sistema en la escuela, deben poder accesarlo desde su casa o desde cualquier lugar.
- *Control de la información.* El *sistema* de animación debe estar parametrizado, es decir, debe permitir a los usuarios crear sus propios conjuntos de datos de entrada para las animaciones.
- *Interactividad.* El sistema de *animación* debe proporcionar interacción entre usuarios y el sistema. La interacción debe contemplar animar paso a paso, hacer un paso o toda la animación, adelantar o regresar la animación a un estado determinado de manera rápida, *etc.*
- *Historia*: El sistema debe *permitir* al usuario regresar y ver estados previos de animación en ejecución.
- *Retroalimentación*: Debe *obtenerse* de alguna manera la retroalimentación de estudiantes para evaluar la efectividad del sistema, de tal manera que sea posible mejorarlo.

El software desarrollado considera algunos de estos lineamientos, sin embargo algunos de ellos faltan por ser considerados y queda como trabajo futuro.

3 Animación del análisis léxico

El análisis léxico es la primera fase de un compilador. Su principal función consiste en leer los caracteres de entrada y elaborar como salida una secuencia de componentes léxicos que utiliza el analizador sintáctico para hacer el análisis. El software aprendizaje que hemos desarrollado cubre la descripción de lenguajes regulares través de expresiones regulares, la teoría de autómatas finitos, la generación autómata finito determinista mínimo a partir de una expresión regular. Todos algoritmos fueron implementados como se describen en [1]. La idea de este software es brindar a los alumnos de los cursos de Compiladores y de Teoría de Autómatas Lenguajes Formales material de apoyo complementario que facilite el entendimiento de modelos abstractos.

Concretamente, la versión actual del software permite especificar una expresión regular (siguiendo la notación descrita en [1]) y a partir de ella genera de manera automática:

- el autómata finito no determinista (AFN),
- el autómata finito determinista (AFD), o

el autómata finito determinista con estados mínimos que acepta el lenguaje descrito por la expresión regular.

Además, el software permite dar como entrada cadenas de símbolos y realizar la simulación automática o paso por paso para ver si la cadena es aceptada o no por el autómata. El software realiza la animación gráfica de los algoritmos de construcción de simulación de los autómatas finitos AFDs y AFNs (ver Fi g. 1).

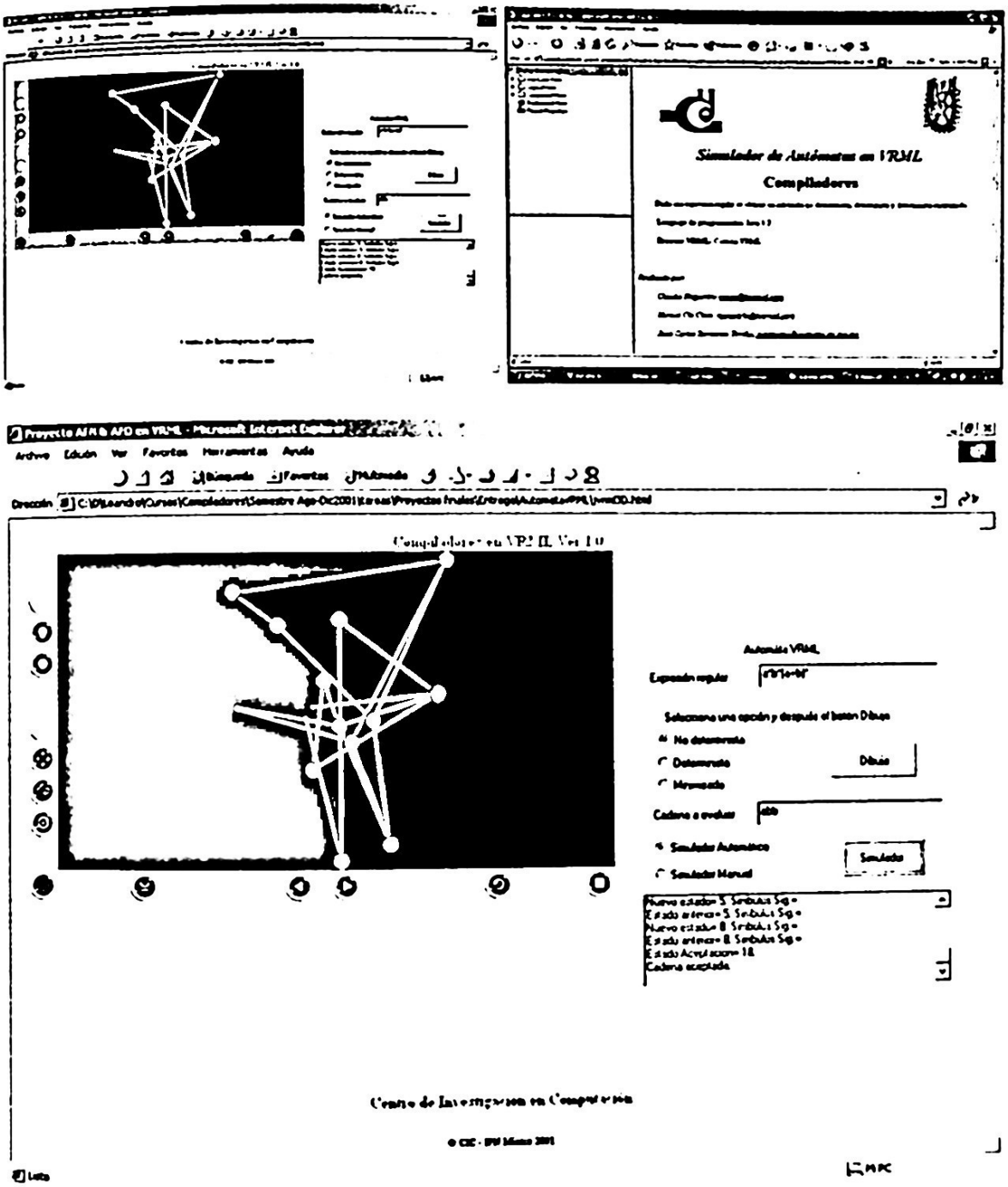

Fig 1. Animación gráfica de la generación y simulación de autómatas finitos.

El autómata que aparece en la Fig. 1 por ejemplo, corresponde a la expresión regular `a*b* (a+b) *`. El autómata que aparece en la figura corresponde al AFN que

acepta el lenguaje descrito por la expresión. En el mundo VRML los estados son representados por esferas grises, el estado inicial por una esfera azul y el final por una esfera roja. Dada una cadena de símbolos, el software permite elegir realizar simulación del autómata manualmente o automáticamente. En el caso de la simulación manual el usuario tendrá que hacer clic sobre el botón Simulador tantas veces como sea necesario hasta que se llegue a un estado final o se rechace la cadena. En el caso de la simulación automática no es necesario ir haciendo clic. En ambas opciones, el software despliega el estado en que se encuentra el autómata así como símbolo de entrada actual en cada paso de la simulación y al final un mensaje indicando que la cadena fue aceptada o rechazada. Durante la animación el estado actual se ilumina de color amarillo, lo cual permite apreciar en que estado se encuentra el autómata en cualquier momento.

4 Diseño e implementación

El software se diseñó e implementó en base al paradigma orientado a objetos. Para el modelado de los diagramas de clases se utilizó la herramienta Rational Rose®. Fig. 2 ilustra el diagrama de clases principal de la aplicación y en la Tabla 1 describe brevemente cada una de las clases que lo componen.

Tabla 1. Principales clases de la aplicación.

Clase	Descripción
`AutomataVRML`	Integra los distintos componentes para llevar a cabo la animación gráfica de los algoritmos.
`Automatax`	Es una clase abstracta que define los elementos comunes un autómata finito: estado inicial, transición, conjunto símbolos, estado final, conjunto de estados.
`Anasin`	Define e implementa el analizador sintáctico para las expresiones regulares.
`Analex`	Define e implementa el analizador léxico para las expresiones regulares.
`MatrizTransAFD`	Define e implementa la estructura de datos para almacenar la descripción del AFD.
`MatrizTransAFDM`	Define e implementa la estructura de datos para almacenar la descripción del AFD mínimo.
`jVRMLaux`	Clase utilitaria para el dibujo de las geometrías que integran el modelo 3D del autómata.

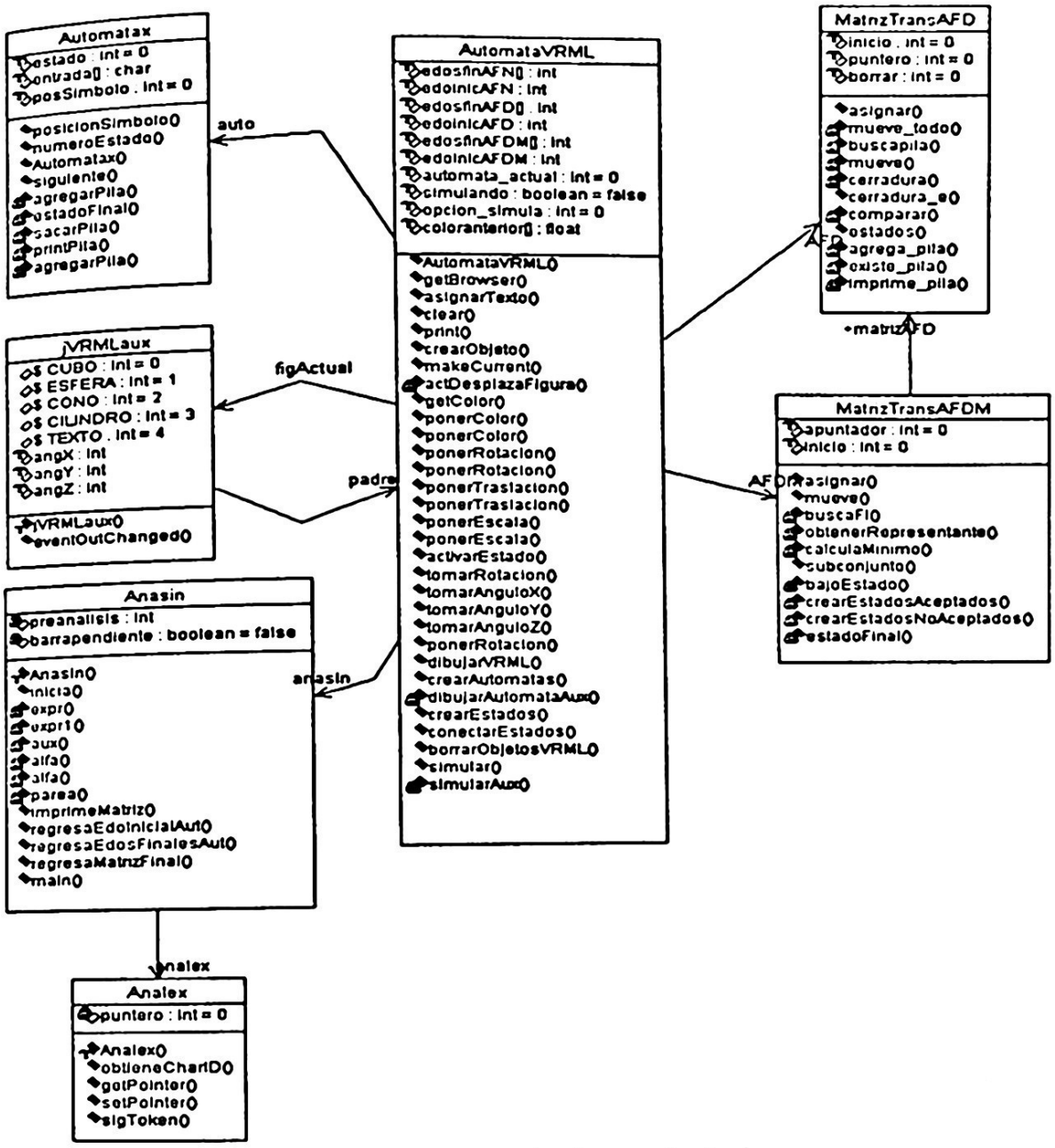

Fig. 2. Diagrama de clases principal.

La utilización de la herramienta Rational Rose para el diseño del software tuvo la ventaja de que permitió documentar el diseño y generar de manera automática dicha documentación en formato Web. Esta documentación ha sido generada con la finalidad de que quienes deseen modificar o extender la funcionalidad del software puedan llevarlo a cabo con relativa facilidad. Se han documentado cada una de las clases, junto con sus atributos, métodos y parámetros de los métodos. La Fig. 3 ilustra parte de la documentación de la clase AutomataVRML y Anasin.

El software es compatible con el Web y para su desarrollo se utilizó los lenguajes Java, VRML [37] y HTML. Java fue utilizado para la implementación de los algoritmos, así como para la implementación del Applet que genera la interfaz 2D. VRML es utilizado para generar la vista 3D de los autómatas y HTML para presentar dicho contenido (ver Fig. 4).

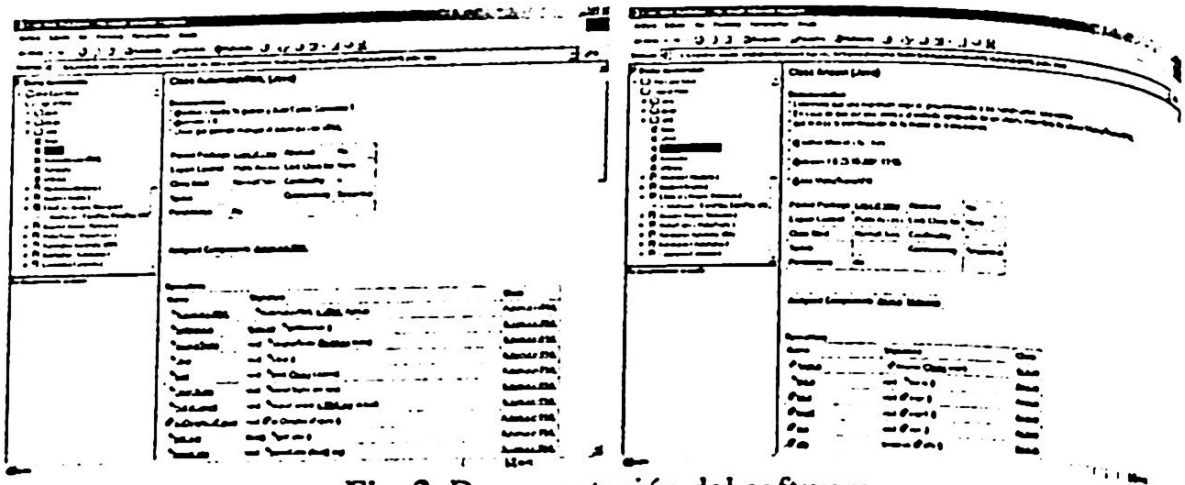

Fig. 3. Documentación del software.

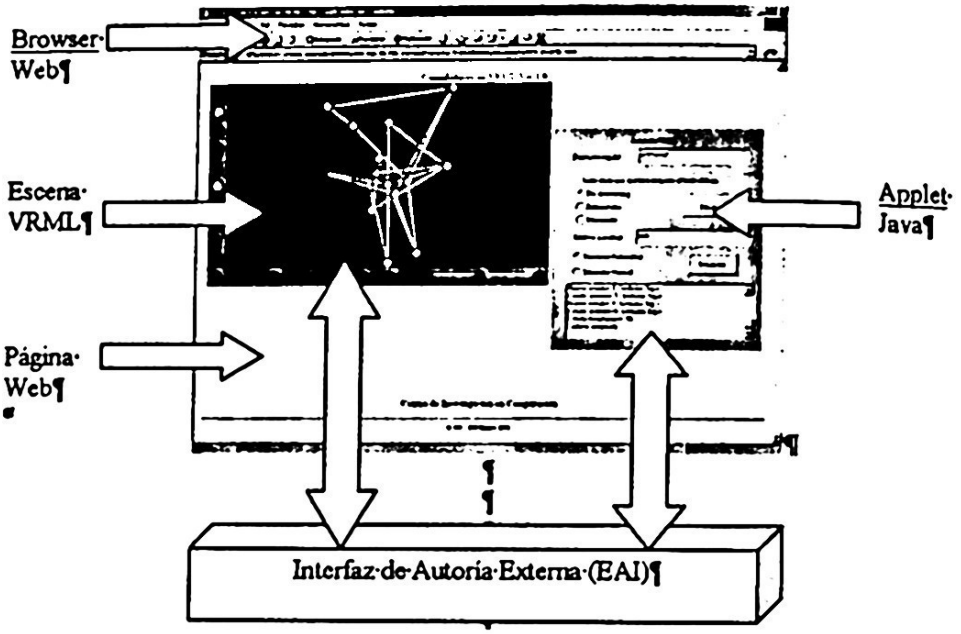

Fig. 4. Tecnologías utilizadas para la implementación del software.

Para lograr la interación del Applet Java con el mundo VRML en el que se construye el autómata, se utilizó la Interfaz de Autoría Interna (External Authoring Interface – EAI) [38]. La EAI es una interfaz de comunicación bidireccional entre VRML y programas externos (por lo general Applets). En nuestro caso, la EAI nos permite modificar dinámicamente el contenido de la escena VRML a través del programa Java; en este caso, el programa Java modifica la escena VRML cuando se generan los autómatas finitos o cuando se realiza la simulación.

6 Evaluación

El software que hemos desarrollado está destinado a estudiantes de los cursos de Compiladores y Lenguajes y Autómatas de Postgrado y Licenciatura. Actualmente esta siendo pre-evaluado por estudiantes del 1er. semestre de la Maestría en Cien-

cias de la Computación del curso de Lenguajes y Autómatas. A pesar de que el software no está terminado al 100%, los estudiantes se han familiarizado fácilmente con las visualizaciones y animaciones y han sido capaces de establecer una conexión de éstas con la teoría (definiciones, algoritmos, *etc.*). Algunos estudiantes han utilizado el software como apoyo para la comprobación de ejercicios previamente realizados con lápiz y papel por ellos mismos, logrando al final un mejor entendimiento de los conceptos y algoritmos. Hemos visto, que la vista 3D generada del autómata resulta atractiva para los estudiantes y éstos han manifestado que les resulta más atractiva esta vista que una en 2D. Además, el hecho de haber utilizado VRML en particular para la generación de la vista 3D agrega al software las capacidades de interacción del visualizador VRML, permitiéndoles a los estudiantes apreciar tanto la construcción como la simulación del autómata desde cualquier punto del espacio tridimensional; los estudiantes pueden rotar el modelo en cualquiera de los tres ejes, acercarse, alejarse, *etc.* Este aspecto ha sido un punto importante e innovador de nuestro software comparado con otros que realizan el mismo tipo de animaciones pero en dos dimensiones. Se ha visto que entre mayor interactividad permita el software a los estudiantes más atractivo resulta para ellos.

Trabajos relacionados

En años recientes, en la Universidad de Saarland en Alemania se han desarrollado visualizaciones en el contexto del diseño de compiladores , incluyendo visualizaciones de máquinas abstractas para lenguajes de programación funcionales, lógicos e imperativos [11], [6], [7]. Estas visualizaciones fueron implementadas en UNIX bajo X-Windows. Las visualizaciones muestran el efecto de la ejecución de instrucciones de máquina sobre el montículo y la pila en tiempo de ejecución, sin embargo, contienen pocas animaciones. Además se ha desarrollado una herramienta llamada VCG (Visualization of Compiler Graphs) para la visualización de grafos que tienen que ver con el diseño de compiladores. Esta herramienta esta disponible para distintas plataformas, incluyendo Microsoft Windows [17], [18], [12]. Además, actualmente existen un gran número de animaciones de algoritmos, sin embargo, se ha desarrollado poco trabajo de importancia en el campo. En [2], [4] se describen muchos de los principios y sistemas existentes para la animación de algoritmos.

En la Universidad de Saarland (Alemania), existe una propuesta similar a la que estamos desarrollando llamada GANIMAL [10]. La meta de este proyecto es crear un software de aprendizaje de exploración para el diseño de compiladores, en el que generen de manera automática a partir de especificaciones las visualizaciones y animaciones de cada una de las fases de un compilador. Sin embargo, los ejemplos que hasta ahora han desarrollado no están basado en el Web y utilizan para su desarrollo una herramienta multimedia llamada ToolBook 3.0 para Windows, a pesar de que se dice que están trabajando en una versión compatible con el Web. Lo anterior hace que la aplicación únicamente pueda ejecutarse sobre plataforma Windows siempre y cuando se cuente con el plugin apropiado para la ejecución del visualizador.

Otros proyectos relacionados al que pretende realizarse son [12], [17], [18], En México, no sabemos de otras Universidades que estén trabajando en proyectos similares.

8 Conlcusiones y trabajo futuro

Hemos desarrollado un software de aprendizaje basado en el Web "Animación Análisis Léxico" que ayuda a los estudiantes a entender mejor los principios construcción de compiladores, en particular de la fase de análisis léxico. El software ofrece, por un lado una introducción interactiva a los problemas de análisis léxico, en el que las definiciones y algoritmos más importantes se presentan de manera gráfica. Las animaciones muestran como es creado un autómata finito a partir una expresión regular, así como su operación.

Una de las aportaciones más importantes de este trabajo en comparación a que realizan el mismo tipo de animaciones, es la utilización de vistas 3D y en particular del uso del lenguaje VRML para la generación de éstas. VRML añade a nuestro software características de realidad virtual, y permite añadirle capacidades interacción en tiempo real adicionales (navegación, caminar, volar, cambiar punto de vista, acercar, alejar, rotar, *etc.*) que lo hacen más atractivo para los diantes.

Aún estamos trabajando en la conclusión del software. Actualmente, estamos bajando en el desarrollo del contenido teórico y las animaciones de algunos de algoritmos de las fases de Análisis Sintáctico y Análisis Semántico. Como trabajo futuro, hemos considerado agregar a nuestro software capacidades multiusuario colaboración, de tal manera que varios estudiantes puedan apreciar la animación los algoritmos e interactuar sobre los modelos al mismo tiempo de manera coordinada . Lo anterior, hará posible la utilización del software para fines de educación distancia. Junto con otros colegas del Centro, hemos desarrollado ideas para construcción de mundos virtuales distribuidos colaborativos basados en el Web [23], que nos permitirán llevar a cabo lo descrito anteriormente.

Referencias

1. Aho, Sethi and Ullman. Addison-Wesley Iberoamericana, 1990.
2. Hausner, D.P. Dobkin. *Making Geometry visible: an Introduction to the Animation Geometric Algoritms*. Computer Science Department, Princeton University, 1996.
3. Braune, S. Diehl, A. Kerren and R. Wilhelm. *Animation of the Generation and Computation of Finite Automata for Learning Software*. In Proceedings of Workshop of plementing Automata WIA'99, Potsdam, Germany, July, Springer Verlag, LNCS 2214. 2001.
4. Brown, Marc H. *Algorithm Animation*. An ACM Distinguished Dissertation: 1987. Cambridge, Massachusetts: The MIT Press, 1987.

5. B. Shniderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* 3er. Edition, Addison-Wesley, 1997.
6. B. Steiner. *Visualization of the abstract Machine MaMa.* Master's Thesis (in German), University of Saarland, Saarbrüken (Germany), 1992.
7. B. Steiner. *Visualization of the abstract Machine WiM.* Master's Thesis (in German), University of Saarland, Saarbrüken (Germany), 1995.
8. Byrne, M.D., Catrambone, R., Stasko, J.T. *Do Algorithm Animations Aid Learning?.* Technical Report GIT-GVU-96-18, August 1996.
9. Duane J. Jarc , Michael B. Feldman , Rachelle S. Heller. *Assessing the benefits of interactive prediction using Web-based algorithm animation courseware.* ACM SIGCSE Bulletin , Proceedings of the thirty-first SIGCSE technical symposium on Computer science education March 2000 Volume 32 Issue 1
10. *GANIMAL Project*, URL: http://rw4.cs.uni-sb.de/~ganimal/
11. G. Kohlmann. *Visualization of the abstract P-Machine.* Master's Thesis (in German), Univeersity of Saarland, Saarbrucken (Germany), 1995.
12. G. Sander. *Visualization of Compiler Graphs.* Technical Report, University of Saarland, Saarbrüken (Germany), 1995.
13. Guido Rößling , Bernd Freisleben. *Experiences in using animations in introductory computer science lectures.* ACM SIGCSE Bulletin , Proceedings of the thirty-first SIGCSE technical symposium on Computer science education March 2000. Volume 32 Issue 1.
14. Guido Rößling , Thomas L. Naps. *Visualisation: A testbed for pedagogical requirements in algorithm visualizations.* Proceedings of the 7th annual conference on Innovation and technology in computer science education June 2002.
15. Hansen, S.R., Narayanan, N.H., Schrimpsher D. *Helping Learners Visualize and Comprehend* Algorithms. Interactive Multimedia Electronic Journal of Computer-Enhanced Learning
16. Herbert L. Dershem , Ryan L. McFall , Ngozi Uti. *Animation of Java linked lists.* ACM SIGCSE Bulletin , Proceedings of the 33rd SIGCSE technical symposium on Computer science education February 2002 Volume 34 Issue 1.
17. I. Lemke. *Development and Implementation of a Visualization Toolkit for Applications in Compiler Construction.* Master's Thesis (in German), University of Saarland, Saarbrüken (Germany), 1994.
18. I. Lemke, G. Sander. *Visualization of Compiler Graphs.* User Documentation, 1994.
19. Kohoe, C., Stasko, J., Taylor, A. *Rethinking the Evaluation of Algorithm Animations as Learning Aids: An Observational Study.* Technical Report GIT-GVU-99-10 March 1999
20. Marc Najork. *Web-based algorithm animation.* Proceedings of the 38th conference on Design automation June 2001.
21. M. H. Brown. *A System for Algorithm Animation.* In Proc. of ACM SIGGRAPH'84, pp. 177-186, July 1984.
22. M. H. Brown. *Algorithm Animation.* MIT Press, 1987.
23. Menchaca R. & Quintero. "*Distributed Virtual Worlds for Collaborative Work based on Java RMI and VRML*", Proceedings of the IEEE 6th International Workshop on Groupware CRIWG 2000, Madeira, Portugal.

24. Peter Gordon McDonald. *Using algorithm animations to assist teaching state space search: an empirical evaluation.* Proceedings of the on Australasian computing education conference December 2000.
25. Quintero-Tellez, R., Menchaca-Mendez, R., Estrada-Segovia. G. M. *Implementation Aspects of Synchronous Collaborative Distributed Virtual Worlds*, 5th IASTED International Conference, Computer Graphics and Imaging, 2002.
26. Rachel Sturm-Beiss , Deborah Sturm. *CS1 concepts using simple animation in Java* The Journal of Computing in Small Colleges , Proceedings of the fifth annual CCSC northeastern conference on The journal of computing in small colleges April 2000. ume 15 Issue 5.
27. Stasko, J.T. *Tango: A Framework and System for Algorithm Animation.* IEEE Computer 23, September 1990, pp.27-39.
28. Stasko, J.T. *Using Student-Built Algorithm Animations as Learning Aids.* ACM SIGCSE, 1997, pp.25-29.
29. Stasko J., Bradre, A., Lewis, C. *Do Algorithm Animations Assist Learning? An Empirical Study and* Analysis. ACM INTERCHI, April 1993, pp.61-66.
30. Stasko J.T., Wehrli, J.F. *Three-Dimensional Computation Visualization.* Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA, Technical Report GIT-GVU-92-20, September 1992.
31. Stephan Diehl. *Specializing Visualization Algorithms.* In F.H. Post, G.P. Bonneau, G.M. Nielson, eds., Proceedings of the Dagstuhl Scientific Visualization Seminar 2001, Kluwer Academic Publishers, May 2002.
32. Stephan Diehl. *Web3D (3D on the Web: From Interactive Simulations to Networked Virtual Learning Environments).* In Handbook on Information Technologies for Education & Training, H.H. Adelsberger, B. Collis, J.M. Pawlowski (Eds.), International Handbook on Information Systems Series, Springer, 2002.
33. Stephan Diehl, Andreas Kerren, and Torsten Weller. *Visual Exploration of Generation Algorithms for Finite Automata on the Web.* In "Implementation and Application Automata", S. Yu, A. Paun (Eds.), 5th International Conference, CIAA 2000, London Ontario, Canada, July 24-25, 2000, Springer Verlag, LNCS 2088, 2001.
34. Stephan Diehl, Carsten Görg and Andreas Kerren . *Animating Algorithms Live and Post Mortem.* In "Software Visualization", State-of-the-Art Survey, Springer LNCS 2269, 2002.
35. Thomas Naps. *Instructional interaction with algorithm visualizations.* The Journal Computing in Small Colleges October 2000. Volume 16 Issue 1.
36. Vic Ciesielski , Peter McDonald. *Using animation of state space algorithms to overcome student learning difficulties.* ACM SIGCSE Bulletin, Proceedings of the 6th annual conference on Innovation and technology in computer science education June 2001. Volume 33 Issue 3.
37. VRML. *Virtual Reality Modeling Language.* URL: http://www.web3d.org/technicalinfo/specifications/vrml97/index.htm
38. VRML-EAI. *External Authoring Interface.* URL: http://www.web3d.org/technicalinfo/specifications/eai/index.html