Universidad Rey Juan Carlos

Sistema Generador de Animaciones Interactivas para la Docencia de Algoritmos Recursivos

TESIS DOCTORAL

Antonio Pérez Carrasco

2011

Universidad Rey Juan Carlos

Sistema generador de animaciones interactivas para la docencia de algoritmos recursivos

TESIS DOCTORAL

Doctorando:

Antonio Pérez Carrasco

Director:

J. Ángel Velázquez Iturbide

2011

Dr. D. Jesús Ángel Velázquez Iturbide, Catedrático de Universidad del Departamento de Lenguajes y Sistemas Informáticos I de la Escuela Técnica Superior de Ingeniería Informática perteneciente a la Universidad Rey Juan Carlos, como director de la Tesis Doctoral “Sistema generador de animaciones interactivas para la docencia de algoritmos recursivos” realizada por el doctorando Antonio Pérez Carrasco

HACE CONSTAR

que esta Tesis Doctoral reúne los requisitos necesarios para su defensa y aprobación.

En Móstoles, a 2 de Noviembre de 2011

Dr. D. Jesús Ángel Velázquez Iturbide

Agradecimientos

En primer lugar, deseo mencionar a mi director de tesis, el Doctor D. J. Ángel Velázquez Iturbide, que ha sabido en todo momento orientar mi trabajo hacia el rumbo adecuado sabiendo alcanzar objetivos relevantes y resultados de interés para el proceso de investigación que ha formado esta tesis. También deseo reconocer la confianza que siempre ha depositado en mí y las oportunidades de desarrollo profesional que me ha ofrecido.

No puedo no aludir a los Doctores Jaime Urquiza Fuentes y Estefanía Martín Barroso, por la ayuda y los consejos tan importantes que me han suministrado en este tiempo y por los que, con total seguridad, seguirán aportándome en el futuro. Es justo mencionar también a los doctores Ari Korhonen y a Lauri Malmi, que me acogieron durante tres maravillosos meses en su país natal, Finlandia, para la escritura de la memoria de esta tesis doctoral, estando disponibles para mí en todo momento incondicionalmente, consiguiendo para mí una estancia muy entrañable e inolvidable.

A continuación expreso mi agradecimiento a los compañeros con los que he compartido risas, horas de trabajo, temores, épocas bajas y momentos impagables: Javier Cano Montero, a quien le debo unos años de carrera inmejorables, Francisco Almeida Martínez, por detallarme con su ejemplo cómo es el compañero de trabajo ideal y a Liliana Patricia Santacruz Valencia, por abrirme nuevas vías de pensamiento y apoyarme justo cuando lo he requerido. Afortunadamente, no sólo los cuento como compañeros, sino también como grandes amigos.

No me olvido del resto de integrantes del Departamento de Lenguajes y Sistemas Informáticos I, todos sin excepción me han aportado antes o después algún valor importante que aprecio profundamente.

Por último, agradezco a mi familia la confianza que siempre han depositado incondicionalmente en mí, apoyándome en mis decisiones y ayudándome siempre que lo he necesitado. Son un pilar fundamental, sin ellos no habría sido posible la culminación de este trabajo.

A todos, siempre y más que nunca, muchas gracias.

Índice General

Índice de Figuras

Índice de Tablas

Resumen

Esta Tesis Doctoral presenta SRec, una aplicación software con fines educativos. SRec es capaz de generar automáticamente y sin esfuerzo por parte del usuario visualizaciones de la ejecución de programas recursivos con múltiples vistas. Éstas ofrecen diversas opciones de interacción y muestran de manera coordinada y coherente la información relativa a un instante de la citada ejecución. La aplicación permite por tanto realizar una exploración a lo largo de sus instantes para ver la progresión de la misma hasta alcanzar el resultado final de la ejecución.

Con este trabajo se pretende ampliar el uso de las visualizaciones en las aulas, no demasiado extendido ante el miedo de los profesores por el consumo de tiempo que puede requerir aprender a usar una aplicación software y por la ausencia de evidencias que demuestren su eficacia educativa.

Para ello se proporciona una aplicación software que supera en cantidad y calidad las características presentadas por otras aplicaciones orientadas a la visualización de la recursividad, aportando un mayor número de vistas, un elevado grado de flexibilidad en su configuración, un mayor número de opciones de interacción y el más completo conjunto de controles sobre las animaciones, proporcionando además algunas facilidades educativas como la exportación del material que se ve en pantalla.

Además, con el fin de facilitar su utilización, la aplicación se ha desarrollado empleando técnicas de diseño centrado en el usuario de manera armonizada con las principales convenios existentes sobre usabilidad. Es por ello que se han realizado exhaustivos estudios sobre usabilidad a lo largo del presente trabajo que garantizan que la aplicación toma progresivamente una forma adecuada para su adopción por parte de los alumnos. También ha sido estudiado el comportamiento de los usuarios mientras trabajan con la aplicación y el modo en que ésta es usada.

El desarrollo de la aplicación también se ha realizado de forma que se permita una fácil ampliación de funcionalidades y vistas gracias a una arquitectura bien definida que consta de elementos totalmente reutilizables. También se ha hecho uso de las tecnologías estándar más extendidas como el lenguaje de programación Java para el desarrollo de la aplicación, el lenguaje de marcado XML para el almacenamiento de datos en disco o los formatos gráficos JPG, GIF y PNG para exportaciones de material

gráfico. A medida que el desarrollo fue progresando, se fueron realizando tareas de difusión para dar a conocer la aplicación.

De esta manera, se presenta un trabajo integral que aúna un estudio del arte, una aplicación software, varios estudios de usabilidad, uso y adecuación de la aplicación, y labores de difusión de la aplicación que culmina en un punto en el que se pretende dotar de mayor profundidad e independencia a diversos trabajos relacionados.

Abstract

This PhD Thesis introduces to a software tool with an educational aim that is able to generate, in an automatic and effortless way, visualizations of the execution of recursive programs. This software tool provides some views that contain several interaction options and show in a coordinated way the information related to an instant of the execution of the algorithm. Users can navigate along the different execution time instants for viewing how it progresses and reach the last and final result of the execution.

This work tries to increase the visualizations usage in the classroom, which is not very extended due to teachers afraid of spending too time learning to use software applications for visualizing, and the lack of evidence about educational effectiveness.

This work provides a software tool that increases the features of existing tools for visualizing recursive algorithms, providing a larger number of views, more flexibility, and a larger number of interaction options and the most complete set of animation controls about animations, providing some educational features as graphical material exportation.

Besides, the development of the tool followed some techniques as user-centered design and main conventions about usability. Along this work some exhaustive usability studies have been made for guaranteeing that the tool takes an adequate form for making easier its adoption by students. The behaviour of the students while they were using the tool and the way the tool is used are aspects that have been studied too.

The design of the tool allows in an easy way the addition of new features, interaction options and views thank to an architecture that uses reusable elements. Other standard technologies as Java (for developing the tool), XML (for saving data on disk) and graphical formats (JPG, PNG and GIF) have been used too.

Some divulgation actions were made while the work was going ahead. A web page was made for letting people know about the tool and download it. A multilanguage support was made for letting the tool be showed in an unlimited number of languages. The tool and its features were introduced into some Congresses and Conferences.

In summary, this is an integral work that joins a study of the state of the art, a software tool, some studies about usability, usage and adequacy of the tool, and some divulgation tasks for the tool. This work culminates in a point that will let start deeper researching lines in some opened works from this work.

Capítulo 1. Introducción

Esta tesis doctoral presenta a SRec, un sistema para la animación de la recursividad con fines docentes que pretende resolver algunas de las carencias existentes en la actualidad a pesar de existir un amplio catálogo de aplicaciones que ya nacieron con la vocación de ofrecer este tipo de funcionalidad.

Algunas de estas aplicaciones ya existentes ofrecen soporte sólo a unos pocos problemas predefinidos [19][97], lo que limita la versatilidad de las aplicaciones. Otras [4][5][12][31][78][94], por su parte, tienen un carácter más genérico, pero, en general, ofrecen un reducido catálogo de acciones de interacción y animación, limitando las posibilidades de un sistema de tales características. Algunas de ellas no suelen ofrecer un número elevado de vistas que dé la oportunidad de complementar la información suministrada por otras.

No sólo se ha tomado como base de inicio el conjunto de aplicaciones existentes con fines similares. También se ha estudiado la representación gráfica de problemas recursivos con carácter general y aquellos diseñados bajo la técnica de diseño “divide y vencerás”, existentes en numerosos libros de texto y manuales de referencia [1][2][6][7][9][11][13][20][32][35][40][71].

Durante este trabajo, se ha seguido de manera cercana cómo SRec es usado y valorado por los alumnos [61][65][67], realizando un profundo estudio sobre cuestiones relacionadas con la usabilidad de la aplicación, evaluadas posteriormente en cinco sesiones de evaluación de usabilidad [70], y también sobre el uso que hacen de la misma los propios alumnos, realizando observaciones sobre el comportamiento y analizando diversos productos obtenidos con ayuda de la herramienta.

Las operaciones de interacción han sido cuidadosamente estudiadas y clasificadas [82]. Una de las más valiosas en la visualización de programas y algoritmos es la animación, para la que se ofrecen varias posibilidades de manejo, siendo, sin duda la principal protagonista durante el uso de la aplicación que se presenta [84][91].

Por último, se han realizado trabajos en evitar que la carga del usuario a la hora de realizar las animaciones sea alta, por lo que con apenas varios toques de ratón se consigue generar una visualización sobre el algoritmo implementado con los valores de

entrada deseados. La creación de visualizaciones sin esfuerzo es uno de los pilares del trabajo técnico realizado.

Se presenta a continuación la motivación de la realización de este trabajo, la hipótesis de partida y las aportaciones de esta tesis doctoral.

1.1 Motivación

La aplicación puede ser utilizada en asignaturas donde se tenga el primer contacto con algoritmos recursivos y en asignaturas de programación más avanzadas donde la algoritmia y concretamente las técnicas de diseño de algoritmos (como “divide y vencerás”) cobran protagonismo.

En las primeras, algunas de las competencias que se adquieren relacionadas más directamente con la algoritmia son:

- Conocimiento de la estructura, organización, funcionamiento e interconexión de los sistemas informáticos, los fundamentos de su programación, y su aplicación para la resolución de problemas propios de la ingeniería.
- Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.

En las segundas, se intentan adquirir, entre otras, las siguientes competencias ligadas a la algoritmia:

- Capacidad para comprender y dominar los conceptos básicos de matemática discreta, lógica, algorítmica y complejidad computacional, y su aplicación para la resolución de problemas propios de la ingeniería.
- Conocimiento y aplicación de los procedimientos algorítmicos básicos de las tecnologías informáticas para diseñar soluciones a problemas, analizando la idoneidad y complejidad de los algoritmos propuestos.
- Capacidad para evaluar la complejidad computacional de un problema, conocer estrategias algorítmicas que puedan conducir a su resolución y recomendar, desarrollar e implementar aquella que garantice el mejor rendimiento de acuerdo con los requisitos establecidos.

En ellas se precisa que el estudiante adquiera los conocimientos y destrezas que le permitan analizar algoritmos, tarea para la que la visualización desempeña un papel importante, al aportar elementos (como la interacción o la animación) que permiten facilitar la ejecución de ese análisis con múltiples fines (entendimiento, estudio de eficiencia, depuración…).

Por otra parte, a pesar de que las visualizaciones en el entorno educativo comenzaron a existir hace tres décadas, no han logrado imponerse en el proceso docente. Esto es debido fundamentalmente a dos causas, relacionadas entre sí:

- No hay evidencias contundentes de que la visualización tenga como consecuencia una mejora en el proceso de aprendizaje [49].
- Alta carga de trabajo para quienes generan las visualizaciones (generalmente profesores) [48].

No extraña por tanto, que ante una falta de eficacia educativa y el elevado coste de implantación, las visualizaciones no sean un recurso empleado masivamente en la enseñanza. Es por ello que, como uno de los objetivos fundamentales, se ha pretendido en todo momento ofrecer un sistema que reduzca el esfuerzo que el usuario (ya sea profesor o alumno) debe realizar para generar visualizaciones, manejar animaciones o interactuar con el sistema en general.

Ciertos autores [27] han adoptado algunos enfoques para reducir el esfuerzo de construcción. Uno de estos enfoques consiste en generar automáticamente visualizaciones altamente enlazadas con el código fuente (visualizaciones de programas); así, cuando un algoritmo es ejecutado, se generan justo a continuación las visualizaciones de los sucesivos estados por los que ha pasado, asociando operaciones y elementos de carácter gráfico a determinadas operaciones. La reproducción secuencial o arbitraria de estos estados de manera interactiva o pasiva es lo que se conoce como animación. Existen evidencias de que las animaciones pueden tener cierta efectividad educativa [81], por lo que queda abierta una vía que permita investigar sobre su posible eficacia con el fin de confirmar o refutar las evidencias existentes.

1.2 Hipótesis

Este trabajo busca mejorar la actividad docente de profesores y el proceso de aprendizaje de los alumnos en asignaturas de informática donde la recursividad sea un concepto importante ayudándose de la propia informática para mejorar la calidad de la docencia.

Parte, en consecuencia, de la hipótesis de que es posible desarrollar aplicaciones software con fines docentes, propiedades de interacción y de carácter automático. A continuación se detallan estos tres aspectos mencionados.

Las aplicaciones con fines docentes son aquellas que asisten a profesores, alumnos o a ambos durante el proceso de enseñanza y aprendizaje, permitiendo complementar y mejorar las clases presenciales, ampliar el catálogo de prácticas posibles para realizar por parte de los alumnos, ofrecer estudios estadísticos de valor para una materia, etc. En el caso concreto de este trabajo, la aplicación servirá para ilustrar gráficamente de manera detallada algoritmos informáticos de carácter recursivo, lo que permitirá mejorar las tareas de comprensión, análisis, construcción y presentación de la recursividad.

Las propiedades de interacción permitirán a los usuarios manejar la aplicación según su conveniencia, para nutrir sus propios intereses, de una manera ágil, sencilla e intuitiva. Sólo así la aplicación resultará útil y no un obstáculo a la hora de usarla para el fin para el que fue creada. Con SRec los usuarios podrán generar tantas visualizaciones como deseen y manejarlas a su conveniencia para realizar los estudios o tareas educativas pertinentes de una manera intuitiva y ágil.

El carácter automático queda concedido por la capacidad de una aplicación de, dados unos datos y/o una orden de entrada, ejecutar automáticamente todas las tareas necesarias sin necesidad de intervención del usuario para lograr el objetivo del mismo. Una vez conseguido, será cuando el usuario podrá manejar los resultados con libertad y flexibilidad gracias a las propiedades de interacción citadas. Para este trabajo, esto se traduce en la capacidad de la aplicación de generar automáticamente las visualizaciones sin que el usuario tenga que intervenir en la creación y puesta en escena de la misma, facilitando el aprendizaje y uso de la misma.

1.3 Objetivos

El objetivo global de este trabajo de investigación es identificar el problema de la implantación de las visualizaciones de programas abordando la recursividad general y manejando casos concretos como el de la técnica de “divide y vencerás”, para aportar una solución mediante una aplicación software orientada a la visualización de algoritmos informáticos que se ajuste a las necesidades de profesores y alumnos.

Como objetivos parciales, se pueden listar los siguientes:

- Desarrollo de una arquitectura genérica para la creación automática de visualizaciones y animaciones de programas recursivos. Esta arquitectura debe ser reutilizable y ampliable. Su estructura global deberá ser capaz de soportar cuantas técnicas de diseño de algoritmos desee incorporar el equipo desarrollador de la aplicación, permitiendo a su vez que las visualizaciones generadas sean flexibles. Ello permitirá ampliar en el futuro la capacidad de SRec con más técnicas de diseño y visualizaciones de una manera muy sencilla, pudiendo ser una labor encomendada a alumnos mediante proyectos de fin de carrera u otros tipos de colaboración.
- Disponibilidad de múltiples vistas de recursividad: esta capacidad permite ofrecer información complementaria sobre diferentes aspectos de la ejecución de los programas, destacando aspectos como la sucesión total de subllamadas recursivas realizadas o las subllamadas que actualmente se encuentran almacenadas en la memoria del sistema.
- Interactividad en visualizaciones. Diversos autores [34][39][49] coinciden en que la interactividad aporta un valor pedagógico a las visualizaciones, por lo que el desarrollo de la misma puede aportar grandes beneficios para la tarea docente que se asigne a una aplicación software.
- Visualizaciones genéricas. En la bibliografía son mayoría las representaciones dependientes del dominio (por ejemplo, un tablero de ajedrez para el algoritmo de n-reinas). Estas tienen mayor carácter expresivo pero una menor capacidad para explicar las técnicas de diseño. Por ello, se pretende construir un sistema que aporte visualizaciones de carácter genérico, basadas en la técnica de diseño y no en el dominio del algoritmo

concreto. El objetivo es construir estas visualizaciones para dos técnicas: recursividad general y “divide y vencerás”.

- Visualizaciones orientadas a la técnica “divide y vencerás”. Manteniendo un enfoque genérico, se pretende ofrecer vistas que permitan complementar a las anteriormente mencionadas para ofrecer aspectos estructurales de los algoritmos diseñados bajo la técnica “divide y vencerás”.
- Facilidades docentes. La aplicación deberá ofrecer, además de las visualizaciones interactivas, de alto valor pedagógico, una serie de facilidades que mejoren y complementen la experiencia educativa, como la capacidad de exportación del contenido del programa a formatos de archivo estándar o determinadas manipulaciones significativas de los elementos para extraer conclusiones de interés más fácilmente.
- Evaluación de la usabilidad. Con el fin de garantizar que se desarrolla una aplicación útil y usable, se pretende evaluar continuadamente entre los alumnos la usabilidad de la aplicación mediante cuestionarios acerca de la opinión general que merece la aplicación así como la capacidad de ilustración de algoritmos y la calidad de diversas partes y funcionalidades.
- Promover la difusión y la adopción de la aplicación. Para ello, deben realizarse actividades como la demostración de la aplicación en congresos y seminarios, la elaboración de una página Web pública que dé a conocer a SRec, promover la facilidad de implantación en cualquier parte del mundo haciendo uso de tecnologías compatibles universalmente y habilitando la opción de que la aplicación esté disponible en múltiples idiomas.

1.4 Aportaciones Principales

A lo largo del tiempo que ha durado el trabajo de esta tesis doctoral se ha logrado un amplio repertorio de publicaciones de distinto nivel de impacto. La Tabla 1 recoge las publicaciones de carácter científico, la Tabla 2 contiene los informes técnicos, la Tabla 3 muestra las aportaciones publicadas en las Actas del Seminario de Investigación en Tecnologías de la Información Aplicadas a la Educación promovido por el grupo de investigación LITE (Laboratory of Information Technologies in

Education), mientras que por su parte la Tabla 4 permite ver las demostraciones y publicaciones divulgativas de SRec.

En todas las tablas los datos se muestran cronológicamente. La primera de las columnas contiene los datos de la publicación (el nombre, el Congreso, Revista o Editorial, y autores), la denominada "C" informa sobre la calidad e importancia de la publicación ("A" indica que es una publicación listada en la categoría A de CORE, "B" indica que es una publicación listada en la categoría B de CORE, "C" indica que es una publicación listada en la categoría C de CORE, "-" indica que es una publicación no indexada en CORE, otras indicaciones están referidas a revistas y editoriales) y la última, los capítulos de esta memoria relacionados con la publicación correspondiente.

Publicación	**C**	**Capítulos relacionados**
A framework for the automatic generation of algorithm animations based on design techniques. EC-TEL 2007, Publicado en LNCS, 2007, Vol. 4753, pp. 475-480. Luis Fernández-Muñoz, Antonio Pérez-Carrasco, J. Ángel Velázquez-Iturbide, Jaime Urquiza-Fuentes.[16]	LNCS	Capítulo 4 Implementación de SRec
SRec: Un sistema para la animación de árboles de recursión. SINTICE 2007. Luis Fernández-Muñoz, Antonio Pérez-Carrasco, J. Ángel Velázquez-Iturbide, Jaime Urquiza-Fuentes.[18]		Capítulo 3 Capítulo 3SRec, Sistema de Animación de la Recursividad
Un sistema con múltiples vistas para la visualización y animación de la recursividad. SIIE 2007. J. Ángel Velázquez-Iturbide, Antonio Pérez-Carrasco, Carlos Lázaro-Carrascosa, Jaime Urquiza-Fuentes.[88]		Capítulo 3 SRec, Sistema de Animación de la Recursividad
SRec: An animation system of recursion for algorithm courses. Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2008). J. Ángel Velázquez-Iturbide, Antonio Pérez-Carrasco, Jaime Urquiza-Fuentes.[91]	A	Capítulo 3 SRec, Sistema de Animación de la Recursividad
A design of automatic visualizations for divide-and-conquer algorithms. V Program Visualization Workshop (PVW 2008) Publicado en ENTCS, 2009, Vol. 224. J. Ángel Velázquez-Iturbide, Antonio Pérez-Carrasco, y Jaime Urquiza-Fuentes.[89]	C ENTCS	Capítulo 3 SRec, Sistema de Animación de la Recursividad
Generación y Mantenimiento de Animaciones de Programas con XML. X Simposio Internacional de Informática Educativa (SIIE 2008). J. Ángel Velázquez-Iturbide, Antonio Pérez-Carrasco, y Jaime Urquiza-Fuentes.[83]		Capítulo 4 Implementación de SRec
Animación automatizada de técnicas de diseño de algoritmos. XV Jornadas de Enseñanza Universitaria de la Informática (JENUI 2009). Antonio Pérez-Carrasco y J. Ángel Velázquez-Iturbide.[63]		Capítulo 3 SRec, Sistema de Animación de la Recursividad Capítulo 4 Implementación de SRec
Aumentando la Interacción con la Visualización de Algoritmos Recursivos. X Congreso Internacional de Interacción Persona-Ordenador 2009. J. Ángel Velázquez-Iturbide y Antonio Pérez-Carrasco.[82]		Capítulo 3 SRec, Sistema de Animación de la Recursividad
SRec 1.2: visualizador integrado de programas recursivos generales y de Divide y Vencerás. XI International Symposium on Computers in Education (SIIE 2009). J. Ángel Velázquez-Iturbide y Antonio Pérez-Carrasco.[84]		Capítulo 3 SRec, Sistema de Animación de la Recursividad
InfoVis Interaction Techniques in Animation of Recursive Programs. Publicado en Algorithms (ISSN 1999-4893), MDPI. J. Ángel Velázquez-Iturbide y Antonio Pérez-Carrasco.[85]	Algorithms	Capítulo 3 SRec, Sistema de Animación de la Recursividad Capítulo 4 Implementación de SRec

Multiple Usability Evaluations of a Program Animation Tool. International Conference on Advanced Learning Technologies (ICALT 2010). Antonio Pérez-Carrasco, J. Ángel Velázquez-Iturbide y Jaime Urquiza-Fuentes.[70]	B	Capítulo 6 Evaluación de Usabilidad con Cuestionarios Capítulo 7 Evaluación de Usabilidad con otros Métodos
Sobre la Interacción en la Visualización del Software. XI Congreso Internacional de Interacción Persona-Ordenador 2010. J. Ángel Velázquez-Iturbide, Antonio Pérez-Carrasco, Jaime Urquiza-Fuentes y Francisco Almeida Martínez.[92]		Capítulo 2 Estado de la Cuestión Capítulo 3 SRec, Sistema de Animación de la Recursividad
Interactive Learning of Recursion by means of Animation. Libro 'Educational Stages and Interactive Learning: From Kindergarden to Workplace Training', IGI Global. Editor: Dr. Jiyou Jia (Peking University, China). J. Ángel Velázquez-Iturbide y Antonio Pérez-Carrasco.[86]	IGI Global	Capítulo 2 Estado de la Cuestión Capítulo 5 Evaluación de Eficacia en Visualización
Experiences in Usability Evaluation of Educational Programming Tools. Libro 'Student usability in educational software and games: improving experiences', IGI Global. Editor: Dr. Carina S. González (Universidad de La Laguna). J. Ángel Velázquez-Iturbide, Antonio Pérez-Carrasco y Ouafae Debdi.[87]	IGI Global	Capítulo 6 Evaluación de Usabilidad con Cuestionarios Capítulo 7 Evaluación de Usabilidad con otros Métodos
SRec as a Plug-in of BlueJ for Visualizing Recursion. VI Program Visualization Workshop (PVW 2011). Antonio Pérez-Carrasco y J. Ángel Velázquez-Iturbide.[68]	C	Capítulo 8 Uso Docente, Esfuerzo del Usuario y Difusión
La Representación de Algoritmos Diseñados bajo la Técnica Divide y Vencerás. XIII International Symposium on Computers in Education (SIIE 2011). Antonio Pérez-Carrasco, J. Ángel Velázquez-Iturbide y Francisco Almeida-Martínez.[69]		Capítulo 2 Estado de la Cuestión Capítulo 3 SRec, Sistema de Animación de la Recursividad Capítulo 5 Evaluación de Eficacia en Visualización

Tabla 1. Publicaciones de carácter científico.

Publicación	**Capítulos relacionados**
SRec versión 1.0: Manual de uso. Serie de Informes Técnicos DLSI1-URJC (ISSN 1988-8074), Vol. 2008-03. Antonio Pérez-Carrasco.[55]	Capítulo 8 Uso Docente, Esfuerzo del Usuario y Difusión
SRec versión 1.1: Manual de uso. Serie de Informes Técnicos DLSI1-URJC (ISSN 1988-8074), Vol. 2009-02. Antonio Pérez-Carrasco.[56]	Capítulo 8 Uso Docente, Esfuerzo del Usuario y Difusión
Resultado de las tres primeras evaluaciones de usabilidad de SRec. Serie de Informes Técnicos DLSI1-URJC (ISSN 1988-8074), Vol. 2009-06. Antonio Pérez-Carrasco, Ángel Velázquez-Iturbide.[60]	Capítulo 6 Evaluación de Usabilidad con Cuestionarios
Resultado de la cuarta evaluación de usabilidad de SRec. Serie de Informes Técnicos DLSI1-URJC (ISSN 1988-8074), Vol. 2010-02. Antonio Pérez-Carrasco, Ángel Velázquez-Iturbide.[65]	Capítulo 6 Evaluación de Usabilidad con Cuestionarios
Quinta evaluación de usabilidad de SRec. Serie de Informes Técnicos DLSI1-URJC (ISSN 1988-8074), Vol. 2011-02. Antonio Pérez-Carrasco, Ángel Velázquez-Iturbide.[67]	Capítulo 6 Evaluación de Usabilidad con Cuestionarios Capítulo 7 Evaluación de Usabilidad con otros Métodos
SRec versión 1.2: Manual de uso. Serie de Informes Técnicos DLSI1-URJC (ISSN 1988-8074), Vol. 2011-05. Antonio Pérez-Carrasco.[59]	Capítulo 8 Uso Docente, Esfuerzo del Usuario y Difusión

Tabla 2. Informes Técnicos.

Publicación	Capítulos relacionados
Herramienta de generación automática de visualizaciones de técnicas de diseño de algoritmos. Actas de I Seminario de Investigación en Tecnologías de la Información aplicadas a la Educación (SITIAE 2007), Luis Fernández-Muñoz, Antonio Pérez-Carrasco, J. Ángel Velázquez-Iturbide, Jaime Urquiza-Fuentes.[17]	Capítulo 3 SRec, Sistema de Animación de la Recursividad Capítulo 4 Implementación de SRec
Visualización de algoritmos implementados bajo la técnica de Divide y vencerás. Actas de II Seminario de Investigación en Tecnologías de la Información aplicadas a la Educación (SITIAE 2008). Antonio Pérez-Carrasco.[54]	Capítulo 3 SRec, Sistema de Animación de la Recursividad
Resultado de las tres primeras evaluaciones de usabilidad de SRec. Actas de III Seminario de Investigación en Tecnologías de la Información aplicadas a la Educación (SITIAE 2009), pp. 143-154. Antonio Pérez-Carrasco, J. Ángel Velázquez-Iturbide.[62]	Capítulo 6 Evaluación de Usabilidad con Cuestionarios
Fundamentos de usabilidad aplicados a un software de fines docentes. Actas de IV Seminario de Investigación en Tecnologías de la Información aplicadas a la Educación (SITIAE 2010), pp. 179-194. Antonio Pérez-Carrasco.[57]	Capítulo 6 Evaluación de Usabilidad con Cuestionarios Capítulo 7 Evaluación de Usabilidad con otros Métodos
SRec como plugin de visualización de algoritmos para el entorno de programación BlueJ. Actas de V Seminario de Investigación en Tecnologías de la Información aplicadas a la Educación (SITIAE 2011), en edición. Antonio Pérez-Carrasco.[58]	Capítulo 8 Uso Docente, Esfuerzo del Usuario y Difusión

Tabla 3. Actas de SITIAE.

Publicación	C	Capítulos relacionados
La Visualización y Animación de Programas como Tecnología para el Aprendizaje de la Programación. I Encuentro de Intercambio de Experiencias en Innovación Docente (celebrado en URJC, 2009). J. Ángel Velázquez-Iturbide, Jaime Urquiza-Fuentes, Francisco Almeida-Martínez y Antonio Pérez-Carrasco.[93]	-	Capítulo 2 Estado de la Cuestión
Interactive Visualization of Recursion with SRec. 14th ACM-SIGCSE Annual Conference on Innovation and Technology in Computer Science (ITiCSE 2009), poster. J. Ángel Velázquez-Iturbide, Antonio Pérez-Carrasco, Jaime Urquiza-Fuentes.[90]	A	Capítulo 3 SRec, Sistema de Animación de la Recursividad Capítulo 8 Uso Docente, Esfuerzo del Usuario y Difusión
SRec, software de animación de la recursividad (recurso docente). XV Jornadas de Enseñanza Universitaria de la Informática (JENUI 2009). Antonio Pérez-Carrasco y J. Ángel Velázquez-Iturbide.[64]	-	Capítulo 3 SRec, Sistema de Animación de la Recursividad Capítulo 8 Uso Docente, Esfuerzo del Usuario y Difusión
Entornos para el aprendizaje visual de la programación. I Jornadas en Innovación y TIC Educativas - JITICE 2010. Antonio Pérez-Carrasco y J. Ángel Velázquez-Iturbide.[66]	-	Capítulo 3 SRec, Sistema de Animación de la Recursividad

Tabla 4. Demostraciones, pósters y otras publicaciones divulgativas.

1.5 Estructura de la Memoria

Se describe a continuación la estructura de la memoria de esta tesis doctoral desde este punto en adelante.

El Capítulo 2, de título “Estado de la Cuestión“, aborda el estado de la cuestión, donde se ofrecerá un enfoque general acerca de la visualización de la información, el

nivel de abstracción más alto, para enfocarse gradualmente en un entorno concreto, el de la visualización de la recursividad, marco en el que trabaja la aplicación desarrollada. Dentro de la visualización de la recursividad se revisarán múltiples modelos, sobre todo gráficos, de representar la recursividad, estudiando su aplicación en obras bibliográficas. En el terreno de la visualización de la recursividad se explorarán diferentes herramientas ya existentes para complementar la visión del punto de partida de este trabajo. Además, se abordará la visualización del software con fines educativos.

El Capítulo 3, titulado “SRec, Sistema de Animación de la Recursividad” presenta la aplicación, de nombre SRec (Sistema de animación de la RECursividad), ofreciendo una completa panorámica sobre ella. Se explorarán las diferentes vistas gráficas que se aportan en las visualizaciones de los algoritmos yse expondrá al proceso de animación de las visualizaciones y las diferentes opciones de interacción. Se comentarán algunas limitaciones de la aplicación (trabajo con código Java procedimental y datos primitivos), inherentes por otra parte a su cometido en ciertos casos.

El Capítulo 4 bajo el título “Implementación de SRec” se adentra en determinados aspectos técnicos de la implementación de SRec como son la apuesta por el lenguaje XML para el almacenamiento de distintos tipos de información (desde datos de procesos internos de SRec hasta visualizaciones del usuario). Se explicará además el tratamiento que hace SRec del código Java inicial que carga el usuario y qué diferencias y peculiaridades tiene el que finalmente ejecuta SRec, pasando por todo el proceso de transformación de dicho código. Además, se realizará una revisión a la arquitectura de la aplicación para observar el uso de patrones de diseño y el modelo arquitectónico que permite a SRec aumentar el número de vistas que se ofrecen sin apenas esfuerzo.

El Capítulo 5, llamado “Evaluación de Eficacia en Visualización”, explora la aplicación desde un punto de vista comparativo con otras aplicaciones existentes cuyo fin es igualmente la visualización de la recursividad. También se discutirá sobre la idoneidad de la aplicación en su estado actual para la representación de la técnica “divide y vencerás”, en el que se hará un repaso de los problemas más comúnmente utilizados para estudiar cómo de útil es SRec para representarlos.

El Capítulo 6 se enfoca en la “Evaluación de Usabilidad con Cuestionarios”, En él se repasan cada una de las evaluaciones de usabilidad realizadas sobre SRec a lo

largo de cinco cursos académicos, revisando las puntuaciones numéricas y las sugerencias aportadas por los alumnos en los cuestionarios que se les suministraba. Además, se efectuará una panorámica global que permitirá ver la evolución de los datos y de la propia herramienta a lo largo del tiempo.

El Capítulo 7, bajo el nombre “Evaluación de Usabilidad con otros Métodos”, evalúa la aplicación acerca de la usabilidad pero sin hacer uso de cuestionarios. En primer lugar, se estudiará en términos cuantitativos el uso que hacen de la aplicación los alumnos. A continuación se comentarán los comportamientos de los alumnos durante la utilización de SRec (si se vieron en la necesidad de emplear papel para hacer dibujos adicionales, etc.) detectados mediante observaciones.

El Capítulo 8, de título “Uso Docente, Esfuerzo del Usuario y Difusión”, recoge aspectos sobre el las tareas relacionadas con la docencia que puede facilitar SRec, el nivel de esfuerzo necesario del usuario y la difusión de la herramienta. Así, se expondrán cuestiones acerca del escaso esfuerzo necesario para la creación, carga y guardado de las visualizaciones y se comentará la posibilidad de internacionalización de SRec gracias al uso del lenguaje XML, que facilita un crecimiento de idiomas ilimitado sin necesidad de recompilar la aplicación. Por otro lado, se aportará una visión general de la documentación prestada a los usuarios, se realizará un recorrido por la página Web oficial de la aplicación, y se presentará, enmarcado como labor de difusión, el plug-in existente de SRec para el IDE BlueJ.

El Capítulo 9, “Conclusiones”, aglomera las conclusiones del trabajo realizado, justificando la satisfacción de la hipótesis y revisando las aportaciones realizadas. Además se ofrece una lista las propuestas de trabajos futuros íntimamente ligados a SRec y a las líneas de investigación que han posibilitado su nacimiento. El Capítulo 10 ofrece los mismos contenidos que el Capítulo 9 pero escritos en lengua inglesa, para cumplir los requerimientos exigidos por la Universidad Rey Juan Carlos para lograr la mención europea del título de Doctor.

Posteriormente puede encontrarse el Anexo con la definición técnica de los documentos XML que genera SRec y los cuestionarios y enunciados de las tareas académicas que realizaron los alumnos en las diferentes evaluaciones de usabilidad de SRec llevadas a cabo.

Capítulo 2. Estado de la Cuestión

Como se ha explicado en el Capítulo 1, esta tesis ofrece una propuesta en forma de aplicación para la visualización de la recursividad con fines docentes. Ésta se enmarca dentro de la visualización de programas, que permite representar gráficamente el cambio dinámico que se produce en la ejecución de los programas siguiendo su código fuente. La visualización de programas queda recogida en un grado de abstracción menor que la visualización de algoritmos, que tiene como objetivo fundamental que los programas sean comprendidos por los humanos haciendo uso de técnicas más o menos complejas de representación. A su vez, la visualización del software es tan sólo una rama de la visualización de información, disciplina que se encarga de convertir datos abstractos en información para ampliar el conocimiento humano a través de la visualización interna de conceptos y datos abstractos.

El concepto de visualización consiste en “la formación de la imagen mental de un concepto abstracto” [28]. Debe notarse que no se menciona la utilización del sentido de la vista ni ningún otro para adquirir la información que ayuda a generar dicha imagen mental, la visualización del concepto en la mente, para ayudar a generar conocimiento. En resumen, la visualización es la creación de una imagen mental a partir de la percepción o tenencia de datos.

El camino habitual para la adquisición de datos son los sentidos, pero en cuanto a información intelectual se refiere, el oído y la vista son las vías fundamentales para la entrada de datos en nuestro cerebro. Nos centraremos de aquí en adelante en la vista, por considerarla de mayor interés y utilidad que el oído para los fines docentes en los que se centra este tesis doctoral. De hecho, la Real Academia Española, aparte de aportar una acepción muy similar a la ya citada, ofrece otras en las que se refiere explícitamente a representaciones gráficas, cuyo camino de introducción de datos en las personas es mediante la visión. La Real Academia Española ofrece las siguientes acepciones para el término “visualizar”:

- Representar mediante imágenes ópticas fenómenos de otro carácter: por ejemplo, el curso de la fiebre o los cambios de condiciones meteorológicas mediante gráficas, los cambios de corriente eléctrica o las oscilaciones sonoras con el oscilógrafo, etc.

- Formar en la mente una imagen visual de un concepto abstracto.
- Imaginar con rasgos visibles algo que no se tiene a la vista.
- Hacer visible una imagen en un monitor.
- Hacer visible artificialmente lo que no puede verse a simple vista, como los rayos X, los cuerpos ocultos, o con el microscopio los microbios.

Las visualizaciones, por tanto, permiten la construcción de los denominados modelos mentales por parte de los alumnos, modelos de conocimiento construidos en base a la información recibida (ya sea textual o gráfica), y que serán mejores cuanto más adecuados sean los modelos conceptuales (representaciones realizadas para enseñar un concepto) expresados por las visualizaciones.

En el apartado “2.1 Visualización de la Información” se profundizará en la visualización de la información, mientras que el apartado “2.2 Visualización del Software” localizará su atención en la visualización del software. El apartado “2.3 Visualización del Software en el Contexto Docente” bajará un peldaño en el nivel de abstracción para explorar la visualización de programas, mientras que el apartado “2.4 Visualización de Programas” entrará de lleno en la visualización de la recursividad y las aplicaciones existentes previamente para tal fin. El apartado “2.5 Visualización de la Recursividad” recorrerá los modelos conceptuales más utilizados en la enseñanza de la recursividad por aplicaciones y profesores y, por último, el apartado “2.6 Modelos Conceptuales de la Recursividad” repasará los conocidos conceptos mentales que los estudiantes suelen crear durante el aprendizaje de la recursividad. De esta manera, a lo largo del presente capítulo se seguirá la estructura representada en la Ilustración 1, desde el punto más externo hasta el más interno.

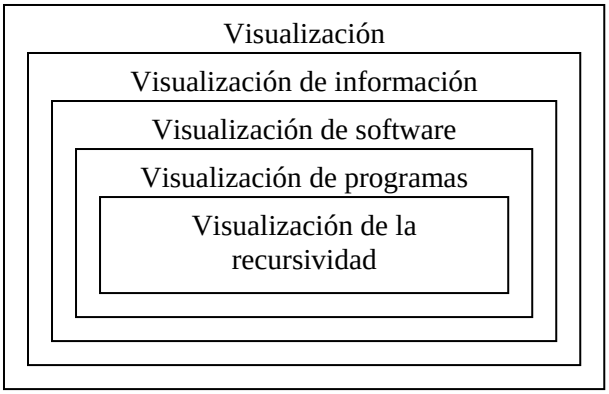


Ilustración 1. Grados de generalidad en la disciplina de la visualización

2.1 Visualización de la Información

La visualización de la información es la disciplina que se encarga de recoger datos abstractos y volcarlos en una representación gráfica que los convierta en información útil con capacidad de informar (previsión meteorológica, mapa de transporte), educar (ciclo del agua) o incluso entretener (pasatiempos). Este manejo y representación de datos suele realizarse mediante un sistema electrónico (ordenador, teléfono móvil u otros dispositivos), si bien podría realizarse de manera manual empleando diversos tipos de materiales.

Los sistemas electrónicos, en los que quedará centrado el resto del capítulo, realizan un proceso de cuatro etapas abiertas (es necesario retroceder a etapas anteriores si los datos cambian) [8] que son:

- Obtención y almacenamiento de los datos: se establece la fuente de la que recogerán los datos (ficheros, sensores, etc.) y se almacenan para que queden disponibles para etapas posteriores.
- Procesamiento de los datos (se filtran los datos de interés y se les da una forma apropiada).
- Interpretación de los datos por algoritmos escogidos (los datos son examinados para empezar a generar la imagen).
- Formación del modelo mental de la información visualizada por parte del usuario (obteniendo así nueva información para él).

A la hora de realizar la representación, se pueden elegir varios enfoques, que ofrecerán resultados muy dispares, como son la representación textual, la gráfica 2D (diagramas) y la gráfica 3D [14]. Se emplean en las representaciones diversas primitivas como texto, puntos, líneas, áreas y volúmenes. Las propiedades que pueden tener son tamaño, longitud, ancho, altura, volumen, posición, orientación, ángulo, color, textura y forma. Se repasan algunas características de la representación textual y de las variantes gráficas 2D y 3D:

- El texto se compone únicamente de palabras, si bien puede servirse de ciertas propiedades (algunas ya nombradas) para realzar algunas partes, como el grosor, el color, el subrayado, la fuente… El arte de presentar texto de un modo visualmente agradable es denominado tipografía.

- Los grafos y diagramas son representaciones gráficas en dos dimensiones que se componen de diversos elementos relacionados geométricamente entre sí, lo cual expresa la relación existente entre los objetos representados mediante tales elementos. Las relaciones geométricas pueden consistir en proximidad espacial, enlazado mediante líneas, contención de elementos en el interior o solapamiento, si bien es cierto que pueden depender de las convenciones usadas y que propiedades como el color, la forma o el tamaño también expresen ciertas relaciones entre elementos de la representación.
- Las representaciones gráficas de tres dimensiones se ayudan de cuerpos con volúmenes, contenidos por caras planas, en forma de triángulos, que son almacenados como una serie de tres puntos y una textura o color que colorea al triángulo. Para poder visualizar los entornos y cuerpos es necesario realizar una tarea de renderizado, que se encarga de calcular los cuerpos visibles en función de la ubicación de la cámara y de ciertas condiciones como la iluminación.

Las representaciones más usadas son los grafos, debido sobre todo a las amplias posibilidades que ofrecen frente a la representación textual y a su extremadamente sencillo manejo y creación en comparación con la gestión de representaciones de tres dimensiones. En ellas existen una serie de normas cuyo cumplimiento da como resultado representaciones más claras y eficaces. Entre estas normas se encuentran, por ejemplo, las siguientes [14]:

- Minimizar el cruce de líneas en el caso de gráficos no planares.
- Minimizar el número de codos en las líneas, deben potenciarse las líneas rectas.
- Minimizar el área total empleada, manteniendo la disposición adecuada de los elementos.
- Minimizar la longitud de las flechas y líneas existentes, sobre todo de las más largas.
- Mantenimiento de las simetrías, todas aquellas que existan entre objetos deben quedar expresadas también entre los elementos que los representan.

- Formación de clústeres, es decir, separar del resto de elementos del gráfico aquellos elementos que están muy ligados, frente a relaciones más débiles (si es que existen) con el resto de elementos.

A menudo los sistemas electrónicos que generan visualizaciones pueden dar pie a visualizaciones dinámicas, que cambian en el tiempo, y que pueden generar animaciones. Éstas reflejan transiciones entre estados o momentos diferentes de un objeto, proceso o concepto abstracto en un orden adecuado y, habitualmente, bajo la interacción del usuario.

Esta interacción de las personas puede permitir realizar acciones muy diversas, como variar la cantidad de información mostrada, el formato de la misma, la disposición… Existen trabajos [98] que clasifican en categorías las posibilidades de interacción:

- Seleccionar: se realza frente a los demás uno o varios datos que tienen un interés o significado especial.
- Explorar: los usuarios pueden revisar el conjunto de datos.
- Re-configurar: se permiten utilizar diferentes perspectivas para mostrar ciertos datos.
- Codificar: produce cambios fundamentales en la representación.
- Abstracción: permite navegar sobre los mismos datos con distinto nivel de profundidad, permitiendo conocer información más concreta o más general.
- Filtrar: los usuarios pueden cambiar el conjunto de datos que se ofrece sin cambiar las condiciones de representación.
- Conectar: se pueden relacionar diversos datos entre sí del total de datos representados.

A través de la visualización de información, por tanto, se consiguen importantes ventajas como:

- Capacidad de expresar una gran cantidad de datos: a menudo las representaciones visuales contienen resúmenes o representaciones básicas de una gran cantidad de información a la que se puede acceder interactuando con la propia representación. Un ejemplo podría ser un mapa digital que

inicialmente ofrece una panorámica mundial: a medida que el usuario se acerca a una región o país concretos, el servicio va ampliando la cantidad de datos que se muestran sobre la región específica, como por ejemplo carreteras o grandes ciudades, mientras que si el usuario sigue acercándose, el servicio empezará a mostrar información mucho más detallada, como posibilidades de transporte público, lugares de interés turístico o económico, nombres de calles. De esta forma, se consigue representar una gran cantidad de datos de interés para el usuario que de manera textual se tardaría mucho más tiempo en asimilar.

- Capacidad de representar relaciones entre datos de distinta naturaleza. Algunos de los tipos de estas relaciones pueden ser:
    - Causalidad o dependencia (secuencias de pasos de procesos)
    - Física (distancias entre elementos).
    - Jerarquía (organigrama de una empresa, red informática).
    - Importancia o representatividad (*tags* de noticias, *newsmap*)
- Representación de patrones y formas esperadas: el hecho de representar datos puede dar lugar a la percepción de ciertos patrones visuales que permitan:
    - Pronosticar tendencias (por ejemplo, predecir la situación en Bolsa de una compañía tomando como base los datos de otra empresa comparable en una situación financiera similar y comparando sus trayectorias)
    - Descubrir errores (notando que ciertos datos no cumplen un patrón establecido o no se mantienen dentro de un rango razonable).
    - Promover mejoras (por ejemplo, comparando la relación entre ingresos y gastos por niveles, departamentos, áreas, etc.).

Sin embargo, no todo son ventajas en la visualización de información. Es posible que en algunos casos la información que se desea mostrar sea muy compleja, o tenga un sentido profundamente abstracto, por lo que no será posible emplear representaciones sencillas o identificar fácilmente qué información se está intentando transmitir.

En ocasiones se pueden representar fácilmente ciertos objetos o conceptos porque existe una relación directa con el mundo real (por ejemplo, el icono de carpeta de los sistemas informáticos para permitir guardar documentos en su interior, a semejanza de las carpetas convencionales físicas), lo que implica que el coste de aprendizaje es prácticamente inexistente. En otras ocasiones se establecen convenios de carácter más o menos universal para determinados conceptos (por ejemplo, el empleo en un mapa de un símbolo compuesto por un cuchillo y tenedor para indicar la existencia de un restaurante); estos no son costosos de aprender, pero requieren cierto esfuerzo inicial. Por último, existen una serie de símbolos creados de manera arbitraria que no responden a ningún origen concreto, pero que son empleados ante la ausencia de una alternativa más intuitiva y familiar para el usuario.

2.2 Visualización del Software

La visualización del software se especializa en el uso de tecnologías electrónicas e interactivas para facilitar la comprensión del software y poder hacer así un uso eficaz del mismo. Puede emplear diversas técnicas:

- Tipografía: se puede emplear para el coloreado de la sintaxis de un código de programación.
- Diseño gráfico: para representar, en base a datos matemáticos, figuras geométricas de menor o mayor complejidad que contengan algún significado o valor.
- Animación: el estado del software varía a lo largo de su ejecución; esa secuencia de estados puede representarse mediante una animación que transite de un estado a otro.

La visualización (de carácter estático) y la animación (con un marcado enfoque dinámico) permiten adquirir conocimiento sobre un artefacto software a través del análisis, la abstracción y la generalización, alcanzando así diversos objetivos como el de la comprensión, la depuración y mejora, o el simple mantenimiento del software que se está visualizando.

El creciente tamaño de los programas y el aumento de su complejidad, hace imprescindible optimizar las técnicas de visualización del software para adecuarlas al

software de hoy día, así las herramientas CASE son uno de los más interesantes ejemplos de adaptación al software existente, pues mediante las diferentes representaciones que realizan (diagramas, gráficas, etc.) permite una mejora del software producido, mayor reutilización del software, mejores soluciones integrales e incluso ahorro de tiempo en las tareas a las que dan soporte. Existe un amplio catálogo de herramientas CASE que se pueden clasificar en libres y de pago, por la plataforma en la que operan (Windows, Linux, Mac, multiplataforma, etc.), la licencia de su código (GNU, código abierto, código propietario) o el enfoque que tienen (ingeniería inversa, modelado de bases de datos, modelado ágil, arquitectura de software, gestión de requisitos…).

Dentro de la visualización del software se pueden encontrar dos ramas diferenciadas. Una de ellas es la visualización de programas, se sitúa en nivel más bajo de abstracción, y visualiza el funcionamiento de un conjunto de sentencias escritas en un lenguaje determinado, por lo que está completamente ligada al código fuente escrito que se visualiza. Además permite establecer convenios de generalidad al representar sentencias tales como bucles y estructuras de control, presentes en gran cantidad de lenguajes. Por otra parte, se encuentra la visualización de algoritmos, que se mantiene en un nivel de abstracción mayor que la visualización de programas al no quedar ligada al código fuente y que ofrece representaciones de algoritmos independientemente del lenguaje de implementación, generalmente disponiendo los elementos necesarios de manera adecuada para la representación del algoritmo concreto que se está visualizando, lo que limita la generalidad de las representaciones.

2.3 Visualización del Software en el Contexto Docente

La informática es una disciplina que, al igual que las ciencias, las artes, las humanidades u otros conocimientos tecnológicos, necesita de labores de difusión y divulgación para sobrevivir al tiempo. En la transmisión de conocimientos en general, y en los de informática en concreto, es habitual la utilización de representaciones gráficas para mostrar de manera esquemática procesos, elementos, cualidades, dependencias, propiedades, composiciones, construcciones y cualquier otro elemento que pertenezca a la disciplina objeto de estudio.

Es por ello fundamental hacer uso de visualizaciones que permitan alcanzar objetivos docentes que la explicación textual no puede conseguir por sí misma. Sin embargo, pese a ser una herramienta fundamental, la visualización no está extendida de manera masiva en las aulas como cabría esperar, y mucho menos con posibilidades de interacción y animación para los estudiantes.

La toma de la decisión de la aceptación de herramientas software de visualización de software, programas o algoritmos recae en la figura del profesor, habitualmente muy reticente a adoptar este tipo de herramientas. Los beneficiados de su adopción son los alumnos, que obtienen la capacidad de realizar más ejemplos, de estudiar por su cuenta con un asistente, etc. A continuación, en los siguientes dos subapartados, se presenta el papel que representan los profesores a la hora de adoptar una herramienta de visualización y las ventajas que tienen para los alumnos este tipo de aplicaciones.

2.3.1 El Papel de los Profesores

La figura del profesor es la que define los conocimientos que se imparten en una asignatura, así como el método docente que se emplea en la misma, ahí es donde cabe la utilización de una aplicación software que complemente las exposiciones teóricas del profesor y que permita a los alumnos tener un papel más activo que la limitación a ver y escuchar. Sin embargo, es muy común que los profesores muestren serias reticencias a incorporar a sus clases aplicaciones software de visualización, de ahí que su utilización aún no esté todo lo extendida que pudiera estar [26].

Los motivos que suelen esgrimirse para la negativa son los siguientes [49]:

- Instalación. El proceso de instalación y configuración inicial suele ser complicado y acarrea por tanto una importante pérdida de tiempo, recurso fundamental y escaso en el contexto docente.
- Aprendizaje. El proceso de aprendizaje puede ser costoso y largo, lo que haría inviable que un alumno lo afronte para emplear la aplicación sólo durante el tiempo que dure una asignatura. Si la curva de aprendizaje es demasiado alta, puede no compensar, ni para profesores ni para alumnos, la adopción de una herramienta.

- Control: miedo a perder el control de lo que sucede en clase, poca confianza en el software. Ante un error de cualquier tipo (pulsar un botón inadecuado, proceso que cuenta con algún error de programación…) el software puede mostrar un comportamiento inesperado que puede bloquear el transcurso de una clase, y esta posibilidad provoca cierto temor entre el profesorado, que prefiere manejar herramientas que conoce más a fondo, aunque estén más limitadas, para evitar adversidades incontrolables.
- Creación: tiempo que requiere la creación de visualizaciones. Si cada vez que se desea realizar un nuevo ejemplo de visualización se pierde una cantidad considerable de tiempo, realmente cabe preguntarse cuál es la verdadera ventaja de hacer uso de un software de propósito específico.
- Mantenimiento. Las labores asociadas a la utilización de un software pueden acarrear también un tiempo elevado. Estas pueden precisar tiempo debido a labores de actualización de la aplicación, aseguramiento de que los laboratorios son compatibles (tanto en hardware como en configuración software) con la aplicación que se pretende emplear, labores de soporte a los alumnos si no saben bien cómo manejar el programa, etc.

También puede añadirse a esta lista de problemas el hecho de que buscar y encontrar una aplicación que se ajuste a las necesidades específicas de un profesor puede ser complicado, pues en primer lugar ni siquiera se tiene la certeza de que tal aplicación exista, y en caso afirmativo, a veces encontrarla o descargarla una vez localizada puede convertirse en una ardua tarea si la accesibilidad no ha sido una cualidad cuidada por los autores de tal aplicación.

Pese a todo, hay profesores que no desisten y que terminan incorporando aplicaciones de visualización en sus cursos. En general, pueden encontrarse cuatro posibles roles entre todos los profesores [3]:

- Profesores que aceptan las herramientas de visualización y las utilizan como base de sus explicaciones en uno o varios temas.
- Profesores que aceptan las herramientas de visualización y las utilizan de manera ocasional y siempre que se ajusten a su manera de dar clase.
- Profesores que no las terminan de aceptar, y que las emplean puntualmente por compromiso (por ejemplo, ante la visita de alguien externo).

- Profesores que las rechazan frontalmente y no las emplean en absoluto.

La falta de evidencias de eficiencia educativa de las visualizaciones, sobre todo con estudios que indican que para conceptos o algoritmos complejos las visualizaciones por sí mismas no aportan ninguna mejoría significativa [34], no ha ayudado a su expansión. Esto ha abierto la puerta a tres hipótesis diferentes [34]:

- Que no existan realmente beneficios significativos del uso de visualizaciones.
- Que sí existan tales beneficios, pero la medida de los mismos no son correctas.
- Que sí existan tales beneficios, pero que algún factor de los experimentos impida que los alumnos se aprovechen de ellos.

Los mismos autores pudieron afirmar que aquellas situaciones en las que los alumnos pueden interactuar libremente para experimentar, analizar y comprender tienen un mayor valor pedagógico que entornos cerrados, donde el alumno se tiene que limitar a realizar una serie dada de procesos o tareas. Además, aunque las visualizaciones por sí mismas no aporten un valor significativo para la docencia, sí logran motivar y atraer al alumno, sobre todo si cuenta con posibilidad de animación e interacción. De esta manera, se logra hacer más accesibles los problemas que desea resolver, de tal forma que sí suponen en última estancia una mejora dentro del proceso de aprendizaje.

2.3.2 Los Beneficios para los Alumnos

Cuando un desarrollador crea un sistema de visualización y un profesor lo adopta para sus clases, los destinatarios de las bondades de esas visualizaciones son los alumnos. Por tanto, conviene conocer también qué papel desempeñan éstos y de qué manera más se pueden beneficiar del uso de visualizaciones.

Los alumnos pueden mostrar dos tipos de aprendizaje [49]:

- Pasivo: los estudiantes actúan como meros receptores de información, escuchando las exposiciones del profesor, leyendo documentación o mirando visualizaciones pregrabadas, sin interacción.

- Activo: los estudiantes toman un papel dinámico, proponiendo y respondiendo preguntas, construyendo y manejando visualizaciones, y, en definitiva, dirigiendo su propio proceso de aprendizaje.

Además, los estudiantes se pueden encontrar ante diferentes escenarios de aprendizaje [14], que requieren la aplicación de uno u otro tipo de aprendizaje:

- Ver animación de algoritmo: si los algoritmos y ejemplos están preestablecidos el alumno adopta un rol pasivo, a no ser que tenga la oportunidad de ejecutar otros ejemplos que él genere.
- Leer algoritmo y ver animación: de nuevo, una lectura sincronizada con el transcurso de la animación lleva a desempeñar el papel pasivo, siempre y cuando el usuario no pueda escoger los valores de entrada ni qué algoritmo visualizar.
- Crear algoritmo y ver su animación: aquí se adquiere un papel activo, este escenario es posible sólo si existen aplicaciones fáciles de usar. Como efecto colateral, el estudiante puede depurar su programa.
- Explorar la estructura funcional: la tarea del estudiante, dado un algoritmo compuesto de múltiples funciones sencillas, es conocer los pasos que da una determinada función. De alguna forma, el estudiante tiene que reinventar el algoritmo, haciendo uso de un rol plenamente activo.
- Prueba del camino visualizado: se pide a los estudiantes que introduzcan ciertos datos de entrada para ver si verifican ciertas condiciones. Gracias a las herramientas de visualización los estudiantes, que representan un papel activo, pueden comprobar la corrección de sus propuestas.

Según múltiples autores, la interactividad es una de las principales características necesarias para dotar de valor pedagógico a las visualizaciones [34][39][49], de tal forma que aquellos estudiantes que empleen el rol activo deberían verse beneficiados del uso de visualizaciones interactivas. Otra importante característica es el texto (escrito o narrado), que debería acompañar a las visualizaciones para explicar los distintos pasos que van teniendo lugar [34][74][79], en cierta manera, esto ayuda a los alumnos más dados a emplear un enfoque pasivo, si bien todos los estudiantes se ven beneficiados.

Los alumnos pueden notar con las visualizaciones los siguientes beneficios [49]:

- Mayor motivación: utilizar este tipo de herramientas hace que aprender resulte más atractivo que empleando un método tradicional.
- Más facilidad de acceso a material práctico: esto permite a los alumnos probar más a fondo los distintos algoritmos y programas, teniendo un catálogo potencial de ejemplos y ejercicios infinito.
- Mejora de la capacidad analítica: el hecho de diseñar casos de prueba aumenta la capacidad analítica del estudiante, mejorando por tanto su proceso de aprendizaje.

2.3.3 Recomendaciones para Mejorar la Eficacia de las Visualizaciones

Para poder conseguir realmente una mejora significativa en el aprendizaje, deben darse una serie de condiciones alrededor de las visualizaciones [72]:

- Facilidad de uso: dado que un estudiante apenas usará un sistema de visualización un par de veces, no va a estar dispuesto a gastar mucho tiempo en aprender a usarlo, por lo que una interfaz compleja arruinaría el trabajo.
- Realimentación apropiada: la realimentación que se proporciona al estudiante debe ir en consonancia con sus niveles de conocimiento.
- Cambios de estado: los diseñadores deben definir claramente los pasos lógicos de las visualizaciones que están programando, incluso a veces es recomendable acompañar los pasos dados durante la visualización con explicaciones textuales.
- Gestión de la ventana: el sistema debería garantizar la visibilidad de todas las vistas para permitir al usuario sacar el mejor partido a la visualización. Además, debería también permitirle disponer las múltiples vistas en la posición deseada y con el tamaño adecuado.
- Múltiples vistas: el hecho de contar con vistas complementarias ayuda a comprender mejor diferentes aspectos de implementaciones, algoritmos y conceptos.
- Control de usuario: el usuario debe poder controlar aspectos como la velocidad de la animación, elegir si se repite cierta parte de la misma o si se

visualiza entera de nuevo, debido a que en diferentes momentos un mismo usuario puede tener diferentes necesidades y velocidad de asimilación, etc.

- Ejemplos predefinidos y valores de entrada: los usuarios deben poder dar valores de entrada para visualizar los ejemplos, pudiendo así experimentar y comprender mejor los programas que están visualizando.
- Pseudocódigo: el pseudocódigo dado en un lenguaje de alto nivel (no en un lenguaje concreto de programación), debe ser claro para permitir al usuario asociar el código del programa con los estados que va visualizando.

A estas se pueden añadir otras recomendaciones [49], muy similares y que coinciden en varios puntos, pero que añaden los siguientes:

- Ayuda para empezar a interpretar las visualizaciones: la primera vez que un estudiante se sitúa delante de una visualización, es posible que no sepa interpretarla completamente, para ello es útil dedicar algo de tiempo a introducir la aplicación o bien añadir textos o narraciones que expliquen lo que se está visualizando.
- Incluir información de rendimiento: la información de rendimiento permitiría conocer mejor la eficiencia del programa o algoritmo que se está visualizando, aspecto que también se puede conocer visualizando a la vez dos algoritmos diferentes que realizan la misma tarea sobre el mismo conjunto de datos de entrada.
- Mantener accesible la historia: que el usuario pueda consultar y repasar los pasos dados con anterioridad puede ayudar a comprender mejor lo que se está visualizando en ese instante.
- Soporte a construcción de visualizaciones: el usuario debe construir sus propias visualizaciones [73], esto le permitirá conocer mejor el algoritmo y qué es lo más importante de él.
- Preguntas al alumno: para animar a la reflexión sobre lo que se está viendo, puede ser interesante la inyección de preguntas en la visualización: bien aleatorias (aunque siempre en un contexto adecuado), o bien ubicadas en un punto específico (y crítico o de especial relevancia) de la ejecución.

2.4 Visualización de Programas

La visualización de programas es un caso particular de la visualización del software. Se enmarca a un nivel más bajo de abstracción, dejando ver la implementación y el funcionamiento de un conjunto de sentencias escritas en un lenguaje determinado, mientras que la visualización del software permanece a un nivel más alto, centrándose en los flujos de datos y resultados obtenidos. La visualización del software puede ser estática (sin animación) o dinámica, donde el usuario puede ver la evolución de un programa. Además, las animaciones de la visualización de programas pueden ser discretas, donde se presenta un número contable de estados o instantes del programa, o continuas, donde no se puede realizar tal distinción.

Existen muchos programas con un enfoque educativo que se orientan a la visualización de programas, dejando ver mediante una animación el transcurso de su ejecución, lo cual permite mejorar su aprendizaje [34].

Kiel [5], WinHIPE [52] y RainbowScheme [31] ofrecen visualizaciones sobre programas pertenecientes al paradigma funcional. Kiel, en concreto, permite visualizar la ejecución de programas de lógica de primer nivel, ejecución que puede ser controlada a través de diversas funciones del programa. La vista muestra los datos con forma de árbol. Por su parte, WinHIPE es un entorno integrado de programación que proporciona una representación gráfica de la evaluación de expresiones como un proceso de reescritura. Permite mostrar la animación paso a paso del proceso que evalúa tales expresiones. RainbowScheme se centra en representaciones semánticas (a través del árbol y la pila) de programas escritos en Scheme.

A su vez, BlueJ [37] y Jeliot [4] están enfocados al paradigma de la programación orientada a objetos, ofreciendo distintas herramientas tales como animaciones, interacción directa con objetos o potentes depuradores. BlueJ se enfoca en la gestión de objetos de una manera visual y sencilla, pues el usuario directamente interactuar con ellos a golpe de ratón para realizar el pase de mensajes, revisar su estado interno, eliminarlos, etc. BlueJ ofrece también a través de su interfaz principal un diagrama UML que deja ver las relaciones entre las clases. Jeliot, haciendo uso de la metáfora de un teatro, produce animaciones de manera automática de programas escritos en lenguaje Java, que pueden hacer uso de estructuras complejas. El programa proporciona además un árbol de recursión.

2.5 Visualización de la Recursividad

La visualización de la recursividad comprende el proceso de representar gráficamente (con posibilidad de interacción y animación) el proceso de recursión, aquel en el que un procedimiento o función informática requiere de su propio servicio una o varias veces para poder encontrar una solución, reduciendo en cada ocasión el tamaño del problema hasta que éste tiene un tamaño que permite abordarlo satisfactoriamente.

Haciendo uso de diferentes modelos conceptuales y métodos de enseñanza, existen una gran cantidad de aplicaciones informáticas que abordan la representación de la recursividad de diferentes maneras, y una de ellas es la animación, muy útil al mostrar los sucesivos estados por los que pasa la ejecución. La representación estática de cada estado se llama visualización.

La interacción del estudiante con la aplicación permite ayudar en tareas de aprendizaje [34], dar soporte a diferentes métodos de enseñanza o proporcionar múltiples tareas educativas. Por ello, la interacción debe poder ser tan amplia como resulte posible. La visualización de programas aplicada en entornos educativos tiene como principal objetivo el análisis de los mismos, para lo que suelen implementarse opciones de interacción como el realzado de determinados elementos o la recogida y resumen de datos.

2.5.1 Aplicaciones Existentes para la Visualización de la Recursividad

En los últimos treinta años han sido desarrolladas, tal y como ya se ha explicado, un elevado número de sistemas para ayudar en el proceso de aprendizaje de la recursión. Se repasan a continuación, diversos sistemas (algunos ya han sido mencionados) cuyo objetivo es visualizar la recursividad. En primer lugar se recorren los que operan dentro del paradigma funcional.

Kiel [5] es un sistema que permite visualizar la ejecución de programas lógicos de primer orden aplicando los métodos de sustitución y simplificación. Soporta parte de Standard ML (SML) y proporciona un amplio número de funciones para controlar la ejecución. La interfaz de usuario muestra una vista con el árbol de la estructura sintáctica de la expresión que se está evaluando, donde pueden mostrarse llamadas recursivas, si existen.

RainbowScheme [31] es un sistema que muestra representaciones visuales con contenido semántico de programas escritos en lenguaje Scheme. Proporciona vistas con el árbol de la ejecución y el estado de la pila basándose para ello en código coloreado. La interacción se realiza mediante iconos y reglas visuales. Gracias a las reglas de transformación de VisualCode [80] el programa puede ser ejecutado paso a paso para ver la ejecución con mayor detalle.

WinHIPE [52] es un entorno integrado de programación que muestra la evaluación de expresiones como un proceso de reescritura. Las expresiones pueden mostrarse in un formato visual para listas y árboles. Además, WinHIPE proporciona un conjunto de opciones de configuración que permiten que la representación sea, según los casos, más entendible y/o agradable en formato. Se puede reproducir la secuencia completa de pasos (o subconjunto de ellos) de la evaluación de una expresión almacenados en un fichero de formato propietario o bien a través de una página Web autogenerada en formato estándar. La animación puede ser guardada y cargada en una sesión posterior para seguir trabajando con ella o, incluso, realizar modificaciones sobre la misma. Un ejemplo de utilización de WinHIPE se puede ver en la Ilustración 2.

La mayoría de los sistemas han sido construidos enfocados al paradigma imperativo. Dentro de estos, se puede distinguir entre sistemas con características visuales para dar soporte a la recursividad y sistemas diseñados específicamente para la enseñanza de la recursividad. En el primer grupo se encuentran dos aplicaciones: ETV y Jeliot.

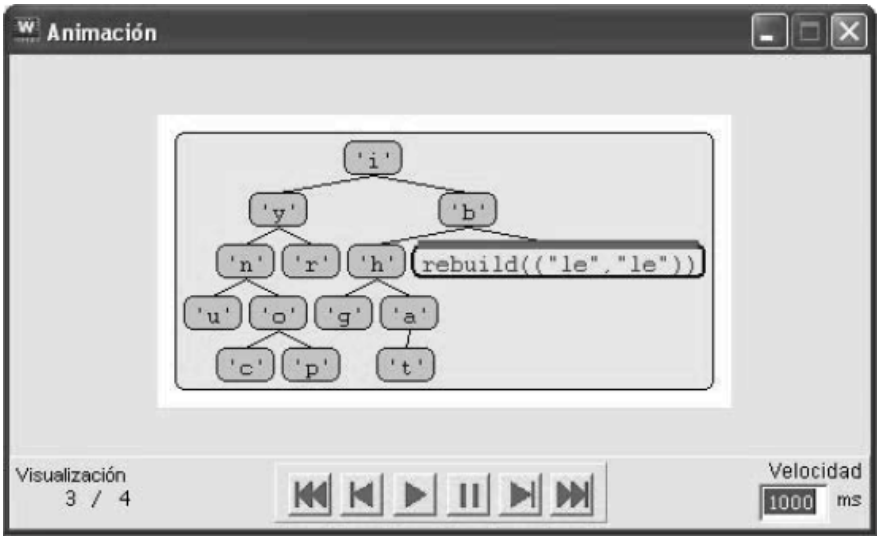


Ilustración 2. WinHIPE ([52], (C) 2007 ACM)

ETV [78] es una herramienta orientada a ayudar a los estudiantes en la comprensión del comportamiento de algoritmos por medio de la visualización de la ejecución de los programas, escritos en C++. La aplicación genera y almacena una traza durante la citada ejecución, que es la fuente de los datos que se muestran en pantalla siguiendo el modelo de copias. ETV, mostrado en la Ilustración 3, ofrece una vista para el árbol de llamadas, otra para valores de algunas variables y otra para el código fuente del algoritmo.

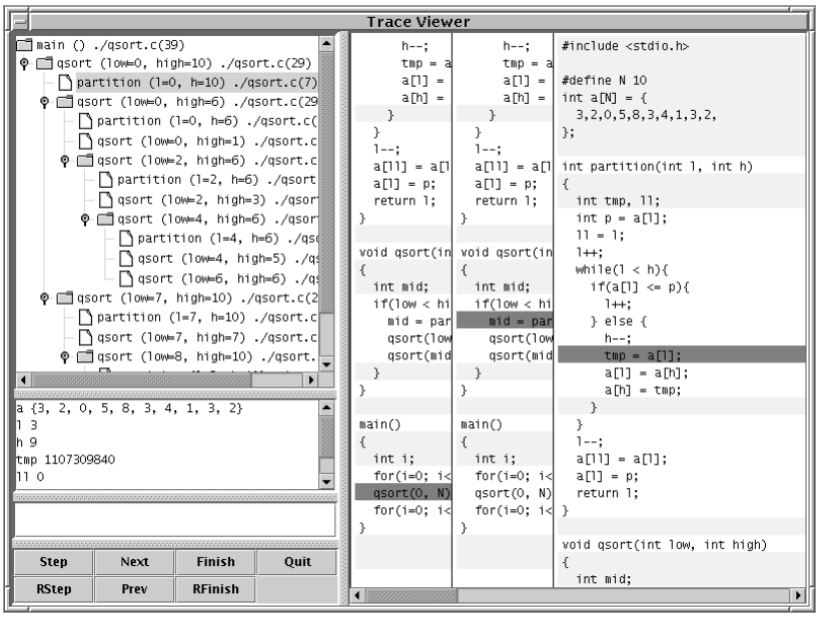


Ilustración 3. ETV ([78], (C) 2005 ACM)

Jeliot [4], por su parte, es un sistema de animación que, haciendo uso de una metáfora de teatro, ofrece visualizaciones y animaciones de la ejecución de programas sencillos escritos en el lenguaje Java donde el usuario puede ver cuántas llamadas están inacabadas en cada momento o cómo se van calculando y almacenando valores. Jeliot proporciona, entre otras características, una vista del árbol de llamadas y una visualización de recursión basada en el modelo de copias, tal y como se puede distinguir en la Ilustración 4.

A continuación se repasan los programas que fueron específicamente diseñados para dar soporte al aprendizaje de la recursividad: EROSI, Flopex 2, Function Visualizer, Recursion Animator y SimRecur.

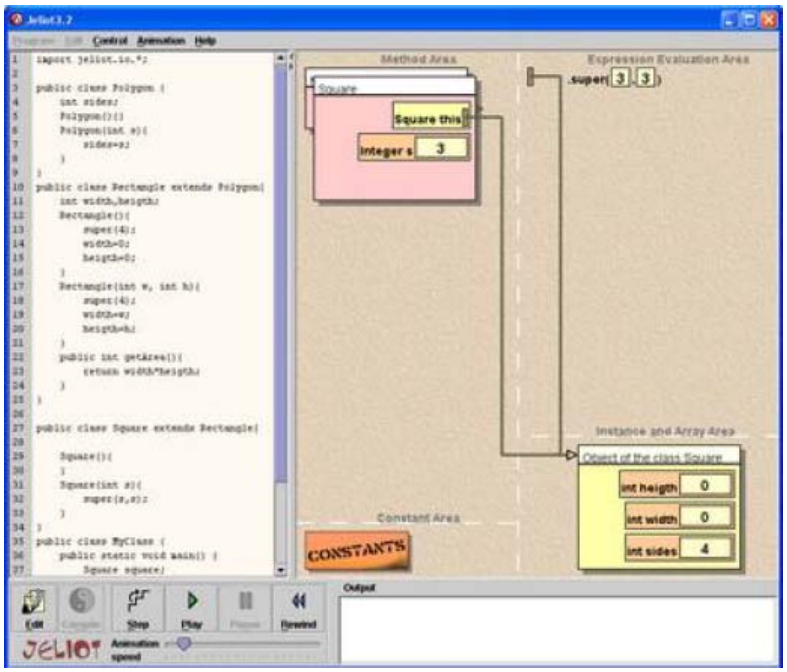


Ilustración 4. Jeliot ([45], (C) 2004 ACM)

EROSI [19] es un sistema de visualización de programas que simula el modelo de copias para expresar la recursividad. EROSI permite a los usuarios observar la ejecución secuencial completa, tanto el flujo activo de control como el pasivo, el flujo de datos y la salida de los programas. Los usuarios se encuentran con un menú en el que pueden elegir varios programas recursivos cuidadosamente ordenados en función de su complejidad. Se puede ver un ejemplo de uso en la Ilustración 5.

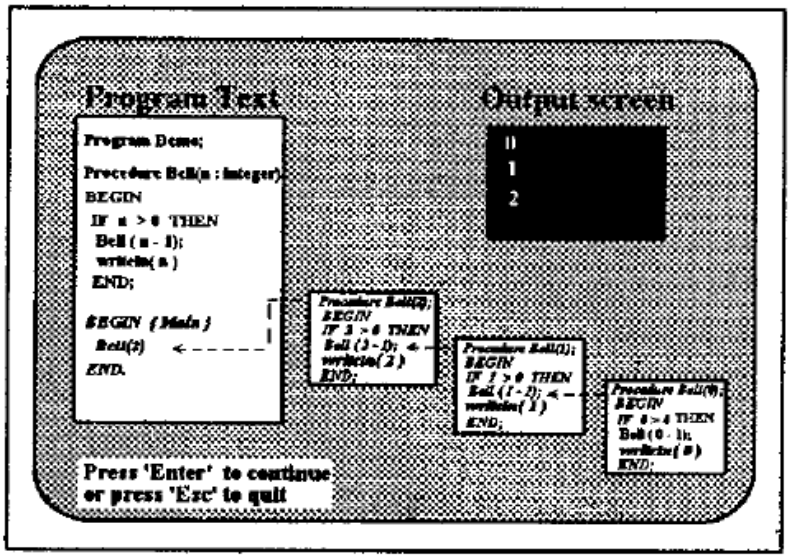


Ilustración 5. EROSI ([19], (C) 2000 ACM)

Flopex 2 [15] es un entorno de programación sobre Microsoft Excel que hace uso de iconos para representar información. Visualiza la ejecución del diagrama de

estados incluyendo en él subrutinas. A lo largo de la ejecución de los algoritmos, se muestran los contenidos de la pila de activación en la hoja de Microsoft Excel donde se encuentra el diagrama. Flopex 2 es capaz también de dibujar un árbol con la historia de llamadas realizadas y las llamadas aún pendientes de finalizar.

Function Visualizer [12] es un sistema que ayuda a los estudiantes en el aprendizaje de cómo se invocan las funciones en el lenguaje Java. Lo que el usuario debe hacer es introducir los parámetros de la llamada inicial. Tras eso, se puede visualizar la ejecución paso a paso, que queda localizada en el código fuente por la iluminación de la línea correspondiente al último paso dado. Cada llamada a una función abre una nueva ventana con su código, por lo que resulta muy sencillo para el usuario saber cuántas llamadas quedan aún por finalizar.

Recursion Animator [94] acepta como entrada cualquier programa escrito en el lenguaje Pascal que tenga al menos una llamada recursiva. Recursion Animator hace uso del modelo de copias, habilitando una ventana nueva para cada llamada recursiva que tiene lugar, donde se pueden ver además valores de variables locales. Estas ventanas se abren cuando se realiza una nueva subllamada recursiva y automáticamente se cierran cuando la correspondiente llamada recursiva finaliza. En la ventana principal se mantiene la traza de la instanciación actual con el código completo de la función. En todo momento el usuario puede cambiar el sentido de la visualización (hacia delante o hacia atrás), lo que permite repasar ciertas partes del algoritmo que pueden no haber sido comprendidas totalmente en una primera pasada.

SimRecur [97] es una aplicación que ofrece múltiples vistas y hace uso del modelo de copias, superponiendo ventanas para expresar las subllamadas recursivas que van teniendo lugar. Las otras vistas muestran la pila de control, donde el usuario puede ver las llamadas pendientes de resolución, sus parámetros y la dirección de retorno, y el árbol de activación, que incluye los parámetros de entrada y los valores de salida.

2.6 Modelos Conceptuales de la Recursividad

Un modelo conceptual es una construcción (textual, gráfica,…) del que el profesor se ayuda para enseñar un concepto. Más formalmente, un modelo conceptual proporciona la representación de un concepto, sistema o fenómeno, habitualmente con

fines educativos, que se considera de mayor calidad cuanto más preciso, consistente y completo resulte para la transmisión de ese conocimiento.

A lo largo de las últimas décadas, los profesores de asignaturas informáticas han propuesto y empleado un gran número de modelos conceptuales [24][96]. El modelo más abstracto es el inductivo, basado en una función o predicado definido en términos de inducción matemática. Por ejemplo, el factorial de un número natural *n* puede ser simbolizado como *n*!, cuyo significado es que $n! = n \cdot (n\text{-}1) \cdot (n\text{-}2) \cdot \ldots \cdot 1$. También se puede expresar recursivamente, tal y como aparece en la Ilustración 6:

$$n! = \begin{cases} 1 & n = 0 \\ n \cdot (n\text{-}1)! & n > 0 \end{cases}$$

Ilustración 6. Expresión matemática del problema del factorial

En esta representación se puede apreciar que existe un caso base, un caso en el que la definición del término no depende de sí mismo (caso de n=0 en el ejemplo), y un caso recursivo, en el que para obtener el valor resultante es necesario hacer uso del propio término que se está definiendo.

Un segundo tipo de modelos conceptuales son las metáforas. Algunas son muy sencillas y están tomadas desde la vida cotidiana, a menudo con elementos naturales o antropomórficos. Habitualmente suelen ser usados en un primer momento, cuando se presenta el concepto de recursión, para posteriormente ser abandonados a favor de otros modelos más elaborados, que permiten profundizar en ejemplos donde la recursividad se torna más compleja. Algunos de los más conocidos ejemplos metafóricos son las curvas fractales [95], los espejos [95] y las muñecas rusas [10].

Éste último modelo expresa de una manera muy clara la recursividad: cada muñeca rusa contiene en su interior muñecas rusas muy similares y de menor tamaño, igual que sucede en los algoritmos recursivos, donde se soluciona un problema pasándole a una llamada recursiva un problema muy similar y de menor tamaño. La última y más pequeña muñeca no alberga ninguna muñeca en su interior, igual que aquella llamada recursiva en la que se alcanza el caso base y no se activa ninguna otra subllamada recursiva más, sino que se soluciona el problema de manera directa.

Otros modelos, como los computacionales, se atienen de manera estrecha a la ejecución de la recursividad. Se proporciona una definición basada en las operaciones

necesarias para calcular el valor del problema, expresadas en un lenguaje de programación, por lo que pueden ser calculadas automáticamente realizando llamadas o invocaciones recursivas.

Algunos de estos modelos computacionales pueden ser usados con cualquier paradigma al ser independientes en ese aspecto. Es el caso de la traza, representación textual de eventos, como las llamadas recursivas y sus resultados obtenidos posteriormente. Todos estos datos, parámetros de entrada y resultados de salida, son escritos en orden cronológico y debidamente indentados para representar con ello el nivel de anidamiento recursivo de cada llamada. Además, pueden emplearse colores para ayudar a diferenciar los dos tipos de datos que se manejan.

No obstante, puede haber casos (por ejemplo, el cálculo del factorial) en los que la traza logra expresar con gran claridad tanto el flujo activo como el flujo pasivo pero otros en los que no (como es el caso de la serie de Fibonacci). Estos casos suelen ser aquellos en los que se da la recursividad múltiple, donde cada ejecución recursiva realiza más de una llamada recursiva a su vez. Para estos casos, está altamente recomendada la utilización de árboles, pues dejan ver de forma clara y evidente la dependencia de las distintas llamadas recursivas. En la Ilustración 7, se pueden ver dos árboles diferentes para el cálculo del cuarto número de la serie de Fibonacci. El primero de ellos, árbol de llamadas, muestra en cada nodo únicamente el valor de los parámetros de entrada; por su parte, el segundo, el árbol de activación, muestra en cada nodo el valor de los parámetros de entrada y el valor de retorno obtenido. En ambos casos, la construcción del árbol se completa secuencialmente en profundidad, desde el lado izquierdo.

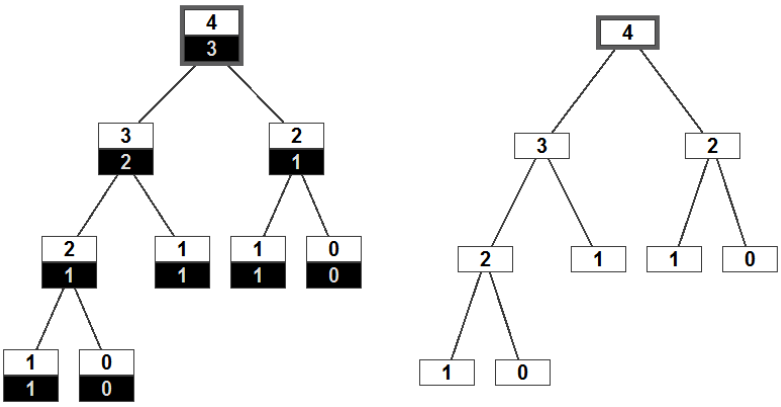


Ilustración 7. Ejemplo de árbol de activación y de recursión

Los modelos del paradigma de programación imperativa dan una visión abstracta de la ejecución del programa en el sistema. Suelen emplearse para ilustrar lenguajes de los paradigmas procedimental, orientado a objetos e, incluso, funcional. Al fin y al cabo, una llamada recursiva no deja de ser un caso particular de la invocación de un subprograma (función, procedimiento, método… serán sus nombres específicos en función del lenguaje).

El modelo de la pila de control (o pila de ejecución) muestra qué llamadas están pendientes de resolver su ejecución. El modelo muestra en qué orden las llamadas se van apilando (activación de la subllamada) y desapilando (fin de su ejecución). En el interior de cada nodo se suelen ofrecer los parámetros de entrada de esa llamada, las variables locales y la dirección de retorno, si bien en ciertas ocasiones, sólo se incluyen los parámetros de entrada. Se muestra un ejemplo en la Ilustración 8. La pila de control está íntimamente ligada al árbol de recursión, ya que su contenido es el mismo que la rama del árbol que liga la raíz del mismo con el nodo activo. Por tanto, se podría decir que el árbol muestra el historial completo de llamadas de la ejecución mientras que la pila se centra en los cálculos pendientes de la ejecución en un determinado instante de tiempo.

Ilustración 8. Modelo conceptual de la pila de control.

El modelo de copias puede ser considerado como una versión ampliada del modelo de la pila de control, en el que se añaden más informaciones como el código del subprograma, desempeñando un rol similar a una “pila de subprogramas”, reforzando la idea de que cada invocación crea una nueva copia de los datos del subprograma. Una representación de este modelo se encuentra en la Ilustración 9:

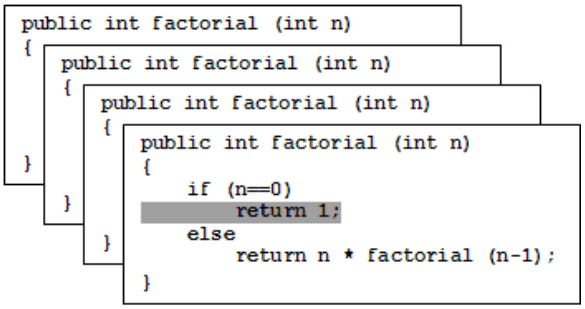


Ilustración 9. Modelo de copias en ejecución

2.7 Modelos Conceptuales específicos para "Divide y Vencerás"

Existen otras representaciones para la recursividad distintas a las ya mencionadas, pero no mantienen un carácter tan general como las presentadas, pues están orientadas al caso específico de la técnica "divide y vencerás". Algunas de estas representaciones dependen del tipo de operación realizado sobre una estructura de datos [75], distinguiendo si se realiza un simple recorrido por una estructura (por ejemplo, una búsqueda), si se manipulan elementos de una estructura (por ejemplo, ordenación) o si se produce una construcción de datos en una estructura (inserción).

Así, un modelo conceptual es el modelo cronológico centrado en una estructura de datos [71]. Este modelo permite ver el estado de la estructura (habitualmente un vector o una matriz) paso a paso a lo largo del desarrollo del algoritmo, lo que permite registrar los diferentes cambios que se van realizando sobre sus contenidos. También, mediante técnicas de coloreado u omisión de valores puede resaltarse qué partes están siendo manejadas en cada subllamada recursiva, lo que permite comprender fácilmente cómo los algoritmos van dividiendo el tamaño del problema manejado y cómo combinan los valores y resultados para obtener la solución. Este modelo, por tanto, ofrece dos tipos de información que resultan de gran interés en algoritmos como la ordenación de vectores o el cálculo de la traspuesta: por un lado permite ver cómo va cambiando el contenido de la estructura y por otro cómo el algoritmo va dividiendo y manejando las partes de la estructura para alcanzar la solución final. Se ofrece una muestra con el algoritmo Quicksort en la Ilustración 10 (a).

Ilustración 10. Modelos Cronológico (a) y de Estructura de Datos (b)

Por otro lado, existe un modelo que igualmente se centra en el contenido de la estructura pero que no está enfocado desde un punto de vista cronológico sino jerárquico de las subllamadas recursivas. Este modelo [75] representa el contenido de la estructura en un estado concreto de la progresión del algoritmo haciendo distinciones entre las partes ya manejadas y las partes que aún están pendientes de ser procesadas por el algoritmo. Se ofrecen tres ejemplos en la Ilustración 10 (estados intermedios en (b.1) y (b.2) y estado final en (b.3) para el algoritmo Quicksort). Además, mediante líneas se realizan separaciones entre las partes manejadas en las distintas subllamadas recursivas basándose en las llamadas recursivas almacenadas en la pila de control, de tal forma que se puede establecer una visión global de cuántas llamadas recursivas están colaborando en la división de la estructura y sobre qué partes se está actuando. Debajo la representación, mediante líneas para vectores y cuadrados para matrices, se representan todas la subllamadas que han tenido lugar hasta ese instante de progreso del algoritmo.

2.8 Modelos Conceptuales de "Divide y Vencerás" en la Bibliografía

Los algoritmos de esta técnica responden al esquema que aparece en la Ilustración 11. Habitualmente ciertas operaciones complejas quedan dentro de la tarea de división o de combinación de resultados, tareas que no pueden ser visualizadas paso a paso por no tener un carácter recursivo y que por tanto, mantiene ciertas lagunas durante el proceso de aprendizaje o análisis.

```
Si (EsCasoBase)
   Resolver
SiNo
   Dividir
   ResolverSubproblemas
   CombinarSoluciones
FinSi
```

Ilustración 11. Esquema de diseño de la técnica "divide y vencerás"

Con el fin de estudiar el empleo de los modelos conceptuales expuestos en el apartado "2.6 Modelos Conceptuales de la Recursividad", se ha repasado un amplio repertorio de bibliografía compuesto de obras de múltiples autores [1][2][6][7][9][11][13][20][32][35][40][71]. Se ha de entender que, antes de la llegada del software de visualización, los autores ya proporcionaban modelos conceptuales de la recursividad y otras técnicas algorítmicas. Se parte también de la hipótesis de que las representaciones empleadas en los libros de referencia son las más extendidas, aceptadas y útiles en el contexto docente. Es por ello que se pretenden comparar los modelos conceptuales de la bibliografía existente con los del software de visualización disponible a día de hoy.

Gracias a esta búsqueda se han podido catalogar hasta 20 problemas diseñados bajo la técnica "divide y vencerás", que tiene a la recursividad como base de su diseño.

Lo que se buscaba, concretamente, era cualquier tipo de representación gráfica empleada para ilustrar, en un sentido general o con un ejemplo concreto, algún concepto o algoritmo, ya fuese de manera parcial o total. Representaciones como diagramas, secuencias cronológicas, elementos esquemáticos o representaciones especiales, encajan en la búsqueda realizada.

Se optó por agrupar los ejemplos de algoritmos encontrados en la bibliografía, lo que permite analizar más cómoda y eficazmente las peculiaridades de los algoritmos a la hora de ser representados gráficamente. Esta clasificación se ha llevado a cabo en

base a criterios estructurales de los problemas, lo que ha permitido establecer la siguiente clasificación:

1. Algoritmos que abordan problemas de carácter matemático, haciendo uso o no de estructuras de datos (por ejemplo, multiplicación de matrices o factorial). Son algoritmos poco propensos a ser representados de manera gráfica, pero en algunos casos sí logran encontrar visualizaciones que ayuden a comprender su funcionamiento satisfactoriamente.
2. Algoritmos que hacen uso de estructuras de datos (vectores, matrices) devolviendo un valor de retorno ajeno a la propia estructura (por ejemplo, un algoritmo de búsqueda maneja un vector y su valor de retorno es un valor entero independiente estructuralmente de dicho vector).
3. Algoritmos que hacen uso de estructuras de datos (vectores, matrices), realizando ciertas modificaciones sobre el contenido de tales estructuras y convirtiéndose éstas en el propio resultado de salida del algoritmo, sin devolver algún valor adicional (por ejemplo, un algoritmo de ordenación, donde el resultado es almacenado en la propia estructura manejada).
4. Algoritmos con un carácter geométrico, manejando puntos en el espacio o incluso polígonos y haciendo uso de propiedades matemáticas para cálculos diversos sobre tales elementos.
5. Otros algoritmos que no encajan en los grupos anteriores.

Ésta es una clasificación que distingue a los algoritmos fijándose en cuestiones muy similares a las que utilizó Stern [75], basada en el tratamiento de datos que se hace de las estructuras de datos suministradas a la entrada, lo que permite definir tres categorías:

1. Recorrido por una estructura (por ejemplo, algoritmo de búsqueda).
2. Manipulación de elementos de una estructura (por ejemplo, algoritmo de ordenación).
3. Construcción de datos en una estructura (por ejemplo, algoritmo de inserción).

Así, las categorías 2 y 3 podrían corresponderse en gran medida con la categoría 3 de la clasificación propuesta propia, mientras que la categoría 1 de esa catalogación se asemeja a la categoría 2 de esta propuesta.

A continuación se repasan las categorías de la clasificación propuesta, explorando las visualizaciones disponibles en libros. Se podrá observar que los problemas de las distintas categorías cuentan con representaciones muy similares en la bibliografía.

2.8.1 Algoritmos Matemáticos

Los algoritmos matemáticos dan una respuesta a problemas que buscan solucionar cuestiones teóricas y de carácter abstracto que provienen desde el campo de la ciencia de las matemáticas. Se reúnen en esta categoría problemas como:

- Multiplicación de enteros de gran tamaño.
- Multiplicación de matrices con algoritmos avanzados.
- Exponenciación.
- Factorial.
- Serie de números de Fibonacci.
- Otros de mayor complejidad (por ejemplo, convolución de funciones y transformada de Fourier).

Todos estos problemas admiten la búsqueda de una solución desde el enfoque de la técnica “divide y vencerás” pero de algunos de ellos no se ha podido encontrar en la actualidad una representación gráfica útil, por lo que la explicación en libros y clases de los mismos se basa en el desarrollo textual y las expresiones matemáticas, haciendo uso así del modelo inductivo. Es el caso de la multiplicación de números grandes [2][6][7][33], de la multiplicación de matrices [2][6][7][32][71], del cálculo de convoluciones de funciones y del de transformadas de Fourier [35][40].

Algunos de los autores consultados hacen uso de sencillos diagramas que buscan situar al lector o explicar propiedades de carácter meramente matemático, como por ejemplo. Algunos ejemplos son la representación estrictamente matemática de la multiplicación de matrices [32][71], o la representación de la multiplicación de números enteros de gran tamaño, escribiendo uno debajo del otro y dejando huecos para el

algoritmo iterativo clásico que se aplica en el cálculo manual [35] y haciendo sobre ellos una primera división en dos mitades que sirva para introducir el algoritmo [6].

Ciertos problemas matemáticos sí admiten una representación gráfica que ayude a su comprensión y análisis. Por ejemplo, el cálculo del factorial, la exponenciación y el cálculo de la serie de los números de Fibonacci tienen como representación sencilla el árbol de activación y/o la pila de control. Sin embargo, de ellos sólo el cálculo del factorial cuenta con representación gráfica en los libros consultados [11] para mostrar el comportamiento de la pila del programa, mientras que para la exponenciación [6][7] y el cálculo de la serie de los números de Fibonacci [6] los autores sólo se apoyan en explicaciones textuales y expresiones matemáticas.

Por su parte, el problema de multiplicación de matrices no puede admitir una representación genérica fácil de diseñar ya que existen numerosos algoritmos que cumplen esta labor ofreciendo unos procedimientos diferentes cada uno de ellos.

2.8.2 Algoritmos que Usan Estructuras y Devuelven un Valor

Estos algoritmos trabajan en todos los casos con una estructura simple de datos, como un vector o matriz, y retornan un valor simple como resultado de su ejecución, que puede ser de cualquier tipo (numérico, booleano…). En este grupo se encuentran los problemas de:

- Búsqueda binaria.
- Máximo de un vector.
- Mínimo y máximo de un vector.
- Selección (búsqueda del *k*-ésimo valor más pequeño).
- Mediana del vector resultante de mezclar dos vectores ordenados.
- Elemento mayoritario de un vector.

El problema de la búsqueda binaria en un vector es representado en la bibliografía explorada a través de dos representaciones principalmente: por un lado se encuentra una representación en forma de árbol binario equilibrado con las posiciones del vector [9], y por otro una representación que indica los pasos que dan los índices que delimitan la parte del vector que maneja el algoritmo en cada llamada recursiva [5]. La segunda es especialmente interesante para poder comprender cómo el algoritmo va

descartando partes del vector basándose en los valores límite de la porción que maneja y en la premisa de que los valores están ordenados.

Por otro lado, y como es lógico esperar, las representaciones de la búsqueda del valor máximo (o mínimo) de un vector y de la búsqueda simultánea del valor máximo y mínimo son muy similares. La más intuitiva es la que contiene el árbol que representa las reiteradas divisiones del vector en un orden descendente y la transmisión de los valores resultantes parciales en orden ascendente [40][71]. La utilización de flechas puede ayudar a entender el sentido en que se desplaza cada elemento de la representación.

El problema de selección, a su vez, es capaz de devolver el valor *k*-ésimo más pequeño de un vector no ordenado basándose en los mismos cálculos de pivote que realiza el algoritmo de ordenación rápida (Quicksort). Este algoritmo cuenta con una representación en libros que refleja en orden secuencial qué partes del algoritmo se van manejando en cada subllamada recursiva y cuáles van quedando descartadas [7]. Además, se separan las posiciones del vector que contienen la posición del pivote y aquellas que dejan de ser de interés para la búsqueda quedan ocultas.

El problema del cálculo de la mediana del vector resultante de mezclar dos vectores ordenados tampoco cuenta con una representación gráfica que especifique cómo se procede durante el algoritmo para encontrar el valor resultante. Por último, el algoritmo que comprueba la existencia de un valor mayoritario en un vector no cuenta igualmente con ninguna representación gráfica en la bibliografía manejada.

2.8.3 Algoritmos que Usan Estructuras sin Devolver un Valor

Este tercer conjunto aglutina a los algoritmos que manipulan una estructura simple de datos, como un vector o matriz, pero que no proporcionan un valor adicional de retorno, pues su ejecución se centra en la modificación del contenido de la citada estructura, cuyo estado final puede ser entendido como el propio valor de retorno del algoritmo en cuestión. Quedan recogidos aquí algoritmos tales como:

- Mergesort (ordenación por mezcla).
- Quicksort (ordenación rápida).
- Coloreado con formas L de un tablero defectuoso.
- Intercambio de dos partes de un vector.

El algoritmo Mergesort concentra la mayor parte de sus operaciones en la tarea de combinación de resultados, no en vano, se dedica a mezclar los resultados parciales obtenidos previamente en las dos llamadas recursivas. Este proceso de combinación no suele representarse en la bibliografía por ser muy fácil de comprender conceptualmente. Para el algoritmo general sí suele hacerse uso de un árbol de activación que refleje el estado inicial y el estado final en cada subllamada [2] así como una sucesión de imágenes que van mostrando cronológicamente cómo va variando el contenido del vector a medida que es ordenado [7][32][71].

El algoritmo Quicksort, por contra, realiza sus principales esfuerzos en las tareas de división del vector, calculando la posición del pivote y dejándolo situado en su posición final, que servirá para delimitar las particiones para las siguientes llamadas recursivas que se generen. Al ser de cierta complejidad conceptual, la bibliografía suele centrarse en desarrollar la representación de esta parte paso a paso para un completo entendimiento. Para ello se representan secuencias que reflejan cómo se van comparando e intercambiando valores y cómo se sitúa el pivote en su posición final haciendo uso de flechas para representar los índices y de distintas convenciones como el coloreado o la doble separación, remarcando así unas partes del vector frente a otras [2][7].

Al igual que ocurre con el algoritmo Quicksort, el problema de colorear un tablero defectuoso con figuras en forma de L concentra en la tarea de división la mayor parte de sus operaciones. Será en ese momento donde se localice el defecto y se coloreen convenientemente los restantes cuadrantes para poder dividir la matriz que se maneja y obtener subproblemas equivalentes al problema original, aunque de menor tamaño. La bibliografía consultada coincide en dibujar la matriz con un elemento en forma de L [71] o con varios [32] para remarcar el proceso de rellenado.

Por último, el algoritmo de desplazamiento de los contenidos de un vector es sencillo de comprender y su concepto suele ser fácil de representar, reflejando mediante alguna separación o diferenciación las partes que se van a desplazar [6]. Además, se puede hacer uso de flechas para representar a los índices que se manejan con el fin de recorrer el vector mientras se mueven los elementos para alcanzar el objetivo final.

2.8.4 Algoritmos de Carácter Geométrico

Esta categoría reúne algoritmos cuyos elementos tienen un carácter puramente geométrico empleando planos, puntos, líneas u otros elementos geométricos. Algunos de los algoritmos recogidos en esta categoría son:

- Cálculo del par de puntos más cercanos entre sí dado un conjunto de puntos en un plano.
- Cálculo del número de puntos que domina cada punto de un conjunto de puntos dados sobre un plano.
- Cálculo de polígonos convexos.
- Cálculo de diagramas de Voronoi.

En general, estos algoritmos son representados de manera directa en los libros [1][2][9][32][35][40][71] utilizando ejemplos donde se sitúan puntos u otros elementos en un eje de coordenadas de dos dimensiones. A estas representaciones se añaden zonas coloreadas, remarcadas, separaciones o bien flechas que sirven para indicar cálculos, relaciones, divisiones del problema, etc.

Para este tipo de problemas, por tanto, predominan las representaciones bibliográficas basadas en el dominio (dominio de la geometría, concretamente) frente a otros algoritmos en los que las representaciones genéricas podían cumplir adecuadamente con una función ilustrativa.

2.8.5 Otros Algoritmos

Aquí se recoge un único algoritmo encontrado en la bibliografía utilizada que no entra en ninguna de las categorías anteriores.

Éste es el problema del cálculo del calendario de un campeonato, que intenta determinar el repertorio de encuentros que deben disputarse los contrincantes de un campeonato. El algoritmo es representado mediante una tabla en la bibliografía consultada [7].

2.8.6 Implantación de los Modelos Conceptuales

En los apartados anteriores se ha recorrido la clasificación realizada de problemas, revisando qué representaciones gráficas se aplicaban para ilustrarlos en las diferentes obras bibliográficas consultadas. A continuación se van a revisar estos

mismos datos pero desarrollando otra perspectiva, la de los propios modelos conceptuales. Se repasa cada uno de los modelos y se detalla para qué problemas han sido usados. Con esto se consigue conocer qué modelos son los más utilizados.

Fueron varios los problemas que se encontraron presentados en las obras desde una perspectiva meramente matemática, haciendo uso del modelo inductivo o de expresiones y notaciones matemáticas. El problema de multiplicación de dos números enteros grandes [2][6][7][35] y el de la multiplicación de matrices [2][7][32][71] fueron aquellos para los que este tipo de modelo fue más utilizado, cuatro veces en total para cada uno. Las convoluciones y transformadas de Fourier se encontraron dos veces [35][40], mientras que para la exponenciación [7] y la búsqueda binaria [6] fue empleado en una ocasión.

El modelo conceptual del árbol de recursión fue empleado en muy pocas ocasiones, en contra de lo que pudiera esperarse al ser una representación de gran expresividad por su completitud y facilidad de entendimiento. Fue localizado en cinco ocasiones, de las cuales dos fueron para ilustrar el algoritmo de ordenación por mezcla [2][35], una para el algoritmo de ordenación rápida [9] (el autor empleó también el modelo cronológico de la estructura de datos para representar este algoritmo), otra más para la búsqueda binaria [9] y una última para representar el cálculo del máximo y mínimo de un vector [71].

La pila fue encontrada en una única ocasión, representando mediante una secuencia el proceso de cálculo del número factorial [11]. Otro modelo conceptual tuvo también escasa utilización, el de la vista de estructura, que fue identificado sólo en la ilustración del problema del coloreado del tablero defectuoso por parte de dos autores [32][71].

Sí está mucho más extendido, por su parte, el modelo conceptual cronológico de la estructura de datos, que fue empleado por tres autores distintos para la ilustración del algoritmo de ordenación por mezcla [7][32][71], y por dos para la ordenación rápida [7][9]. Se identificó este modelo también en la ilustración de los problemas de la búsqueda binaria [7], la búsqueda del *k*-ésimo elemento más pequeño de un vector [7], la búsqueda del máximo de un vector [40] y el intercambio de dos partes de longitud variable de un vector [6].

El modelo de la estructura, por otra parte, permite ver el estado de la misma (es decir, cómo ha sido dividida y qué valores contiene) en uno o múltiples instantes de la ejecución, de tal forma que se presenta en una única representación la forma en que el algoritmo maneja la estructura y también sus datos. Ha sido encontrado para visualizar, por ejemplo, el problema del calendario de una competición o el del coloreado de un tablero defectuoso [71].

El modelo geométrico sirvió para abordar todos los problemas que fueron identificados anteriormente dentro de la categoría de algoritmos geométricos. Este modelo fue empleado en cinco ocasiones para representar el problema de hallar el par de puntos más cercanos, dado un conjunto de puntos situados en un plano [2][32][35][40][71]. Además también sirvió para representar otros problemas como el cálculo de los puntos dominados por cada punto [40], el cálculo del polígono convexo [40], la creación de diagramas de Voronoi [40] y, curiosamente, el cálculo de la mediana (k-esimo valor más pequeño) de un vector [2], dándole un aspecto más abstracto al problema. Este enfoque, orientado al dominio, permite abarcar una amplia variedad de problemas que tengan siempre una base matemática y espacial.

La utilización de sólo texto sin ningún recurso visual fue llevada a cabo sobre varios problemas: ordenación rápida [6][71], la búsqueda del *k*-ésimo valor más pequeño de un vector [6][71], la ordenación por mezcla [6], la búsqueda binaria [2], la exponenciación [6], la multiplicación de matrices [6], la serie de Fibonacci [6], las convoluciones de funciones [6], la transformada de Fourier [6], y la búsqueda del elemento mayoritario de un vector [6].

Quedaron sin identificar en ningún caso tres modelos conceptuales muy importantes: el de metáforas, el de la traza de ejecución y el de copias. El modelo de metáforas es, quizá, un modelo muy apropiado para introducir el concepto de recursividad de una manera genérica, pero pierde fuerza rápidamente al ser extremadamente sencillo. No obstante, no es complicado crear visualizaciones empleando este modelo para explicar ciertos algoritmos, como el de las muñecas rusas en aquellos casos en los que no se da la recursividad múltiple, como el cálculo del número factorial.

Por otra parte, el modelo de la traza de ejecución tampoco fue encontrado, seguramente por haber encontrado en otros casos un modelo más expresivo. El modelo

de copias fue otro de los ausentes, si bien no es de extrañar, pues resulta extremadamente costoso en espacio representarlo en un medio físico como el papel. Esto es debido a que, para que el modelo de copias sea eficaz, es necesario ver cómo se comporta a lo largo de toda una ejecución, lo que se traduciría en una secuencia de representaciones ocupando una gran cantidad de espacio en la publicación o libro correspondiente. Probablemente la falta de espacio haya sido una de las causas de su no utilización en las obras exploradas, de lo que se puede extraer que resulta mucho más adecuada su utilización en aplicaciones informáticas que en un medio físico como el papel.

En la Tabla 5 se listan los modelos conceptuales de los que se ha hablado, repasando qué algoritmos han sido ilustrados haciendo uso de ellos y en qué categoría los habíamos clasificado.

La categorización realizada de los algoritmos, como ya se ha explicado, da lugar a cinco clases y, en cierta medida, cada modelo conceptual es empleado sólo por algunas de ellas. El modelo inductivo y matemático es utilizado casi exclusivamente para ilustrar los algoritmos de la categoría de problemas matemáticos, lo cual cuenta con especial sentido. Este modelo, que es el más recurrido junto al textual, es empleado también anecdóticamente para la ilustración del algoritmo de búsqueda binaria.

El modelo del árbol, el tercero menos usado, es usado sólo en algunos casos por algoritmos de las categorías 2 y 3 (algoritmos que emplean una estructura de datos tal como un vector o un array, para devolver un resultado o realizar cambios en la propia estructura, respectivamente). Además, se da la circunstancia de que en todos los problemas en los que se utilizó, otros autores optaron por representar tales problemas con modelos conceptuales diferentes, sobre todo con el modelo cronológico de la estructura.

El modelo de pila, como ya se ha indicado, fue únicamente encontrado para ilustrar un problema, que pertenece a la clase de problemas matemáticos. Este modelo es uno de los más ligados a la manera de trabajar de los ordenadores, pues expresa de manera directa el proceso de apilamiento de subprogramas en la memoria principal.

Problemas \ Modelos conceptuales	**Inductivo/Matemático**	**Árbol**	**Pila**	**Cronológico de la estructura**	**Estructura de Datos**	**Orientado al dominio (Geométrico)**	**Texto**
Multiplicación enteros grandes	4						
Multiplicación de matrices	4						1
Exponenciación	1						1
Factorial			1				
Serie números de Fibonacci							1
Convolución y Transformada Fourier	2						1
Búsqueda binaria	1	1		1			1
Máximo y/o mínimo de un vector		1		1			
Selección				1		1	2
Mediana de vector mezcla de 2 vect.							1
Elemento mayoritario de vector							1
Mergesort		2		3			1
Quicksort		1		2			2
Coloreado tablero defectuoso					2		
Intercambio de partes en un vector				1			
Par de puntos más cercanos en plano						5	
Puntos dominados en un plano						1	
Polígonos convexos / Diagr. Voronoi						1	
Campeonato					1		
Total de utilización	12	5	1	9	3	8	12
Porcentaje de utilización	24%	10%	2%	18%	6%	16%	24%

Tabla 5. Modelos conceptuales usados en la bibliografía.

El modelo cronológico de la estructura de datos es empleado para ilustrar problemas de las categorías 2 y 3 (algoritmos que emplean una estructura de datos tal como un vector o un array, para devolver un resultado o realizar cambios en la propia estructura, respectivamente), igual que ocurría con el modelo del árbol, si bien consigue un mayor uso que ese modelo. Éste es el tercer modelo conceptual más usado, por detrás del modelo inductivo y matemático y de la explicación textual.

El modelo de la estructura de datos, con un uso más testimonial (no en vano, es el segundo menos usado), es empleado donde la solución se aporta en forma de tabla, los problemas en los que fue usado pertenecen a las categorías 3 (problemas que

manejan estructuras sin aportar un valor adicional de retorno) y 5 (compuesta por el problema de la competición).

Algo parecido a lo que ocurría con el modelo conceptual inductivo y matemático sucede con el geométrico. Aquellos problemas que fueron catalogados en la clase 4 (problemas geométricos) fueron ilustrados con este modelo conceptual, que también se empleó anecdóticamente para el problema de la selección del *k*-ésimo elemento más pequeño de un vector no ordenado. Este modelo es el único orientado al dominio encontrado.

Por último, el modelo textual, sin representación gráfica, fue empleado en doce ocasiones para ilustrar diez problemas diferentes en total. De esos diez problemas, tres no contaron con ninguna representación gráfica en la bibliografía consultada.

Los problemas en los que al menos un autor sólo utilizó texto para presentarlo son: multiplicación de matrices, exponenciación, serie de números de Fibonacci, convolución de funciones, transformada de Fourier, búsqueda binaria, selección, elemento mayoritario de un vector, mediana del vector mezcla de dos vectores, ordenación por mezcla y ordenación rápida.

La utilización de sólo texto para la explicación de un problema permite extraer dos conclusiones:

- Si la buscó, el autor no encontró ninguna representación gráfica que fuera satisfactoria para expresar las características del algoritmo.
- El autor consideró suficiente la utilización de texto para poder exponer el algoritmo eficazmente.

Tal y como se refleja en la Tabla 5, los modelos conceptuales más empleados son los dos que menos uso realizan de la representación gráfica, pues el modelo inductivo y matemático se limita a explicar con expresiones formales las soluciones a los problemas mientras que el texto se limita al uso de palabras para hacerle llegar al lector el concepto. Ambos alcanzan el 48% de los problemas revisados dentro de la bibliografía consultada, por lo que la representación gráfica está presente en el 52% de las ilustraciones, poco más de la mitad.

La posible falta de espacio según el tipo de obra, el voluntario interés por ofrecer una exposición escueta, la no existencia de una visualización satisfactoria o el

convencimiento de que el texto es suficiente o la mejor vía para la exposición de los problemas son algunos de los posibles motivos que pueden haber provocado el amplio número de presentaciones sin representación gráfica.

2.9 Modelos Mentales de la Recursividad

Un modelo mental define la representación cognitiva de un conocimiento. Cada persona puede construir un modelo conceptual único para cualquier concepto, por eso no es de extrañar que la explicación que se da para definir qué es la recursividad sea muy diferente entre alumnos y profesores [41] e incluso entre los propios alumnos entre sí [42].

La investigación en este campo ha permitido encontrar errores de comprensión entre los estudiantes. En condiciones ideales, un modelo mental debería tener las mismas cualidades y propiedades que un modelo conceptual; es decir, debería ser preciso, consistente y completo. Como los modelos mentales son construcciones de los propios estudiantes, resulta imposible determinar si es correcto o no, pero sí se puede afirmar si es viable o no. Un modelo mental es viable si permite completar satisfactoriamente una tarea del tema de estudio, y no lo es en caso contrario.

Conocer qué errores de comprensión se producen entre los estudiantes puede ayudar en gran medida a mejorar la enseñanza. A través de estudios realizados con lenguajes de bases de datos se han estudiado estos errores y se han podido identificar algunos modelos no viables [33]:

- Modelo de bucle: el subprograma recursivo es visto como un objeto simple sobre el que se realiza una especie de proceso de iteración.
- Modelo sintáctico: se reconocer elementos sintácticos como indicadores de recursión y se tiene cierta ida sobre su comportamiento, pero sin llegar a un entendimiento completo.
- Modelos raros: los mantienen estudiantes con ideas peculiares sobre construcciones de programación.

Existen estudios exhaustivos de investigación como el de Götschi et al. [21] empleando un lenguaje de programación estándar, en los que se analizaron las respuestas de dos preguntas de un examen dos años consecutivos. Posteriormente, se

volvieron a insertar tales preguntas en una nueva ocasión para confirmar y afianzar los resultados. Estos permitieron identificar varios modelos mentales tales como el modo activo, el modelo de paso, el modelo de valor de retorno, y el modelo algebraico, entre otros.

El modelo activo es similar al modelo de copias, pero sin incluir el control de flujo pasivo, mientras que aquellos que construyen el modelo de paso no conciben los flujos de control, simplemente ejecutan la estructura “if-else”. El modelo algebraico contempla el programa recursivo como un problema algebraico. Todos ellos lograron ser deducidos en función de las soluciones dadas a las preguntas del examen, dejando ver diferentes carencias en la asimilación del concepto de recursión.

Este estudio permitió ordenar ciertos modelos mentales de acuerdo a su viabilidad. Así, tras el modelo de copias, el modelo activo y el modelo de bucle son viables en ciertos casos para algoritmos recursivos finales. Aquellos alumnos que construyen el modelo sintáctico deberían reforzar sus conocimientos, pues es un modelo no viable en múltiples ocasiones. Otros modelos no viables requieren la intervención del profesor para corregir el proceso de aprendizaje, como son el modelo raro, el modelo de paso, el algebraico y el modelo de valor de retorno.

También existen evidencias gracias a este estudio de que los estudiantes suelen hacer uso de más de un modelo mental según la situación concreta y dependiendo de qué modelos son viables. Götschi et al. [22] reforzaron su forma de enseñanza y comprobaron que en general los estudiantes resultan beneficiados si se ven expuestos adecuadamente desde el principio a algoritmos complejos y que hagan un uso importante del flujo de control pasivo, lo que desecha la metodología de comenzar con ejemplos muy sencillos, como el cálculo del número factorial.

Existen catalogaciones de modelos mentales de recursión [44] pero no dejan de ser muy similares a las de los autores del estudio mencionado.

2.10 Conclusiones del Capítulo

En este capítulo se ha presentado el concepto de visualización de la información, que recopila datos de diversa naturaleza para representarlos gráficamente, y se ha profundizado en la rama de la visualización de software, que se encarga de crear representaciones gráficas de programas informáticos con el fin de comprenderlos,

abriendo así la puerta a labores de mantenimiento (adaptativo, correctivo, evolutivo o perfectivo).

Se estableció la diferencia entre visualización y animación (representación estática frente a representación dinámica) y se presentaron las ramas de visualización de programas, con representaciones de carácter genérico y bajo nivel de abstracción (ligadas al código fuente) con la visualización de algoritmos, con representaciones generalmente dependientes del dominio o adecuadas a las características concretas de un algoritmo y de mayor nivel de abstracción (no ligadas al código fuente).

También se exploró la visualización de software con fines docentes, encontrando una reticencia de la comunidad docente por incorporar aplicaciones software de visualización debido a los costes de instalación, aprendizaje, creación y mantenimiento, así como al temor a perder el control de las clases mientras se utilizan las aplicaciones.

La falta de evidencias de eficacia educativa no ha ayudado a este fin, aunque sí existen indicios de que la interactividad logra que el alumno se implique más. Los alumnos, que pueden adoptar roles activos o pasivos, se verán más beneficiados por el uso de estas aplicaciones cuanto más activos sean durante las clases. En general, con este tipo de aplicaciones los alumnos pueden estar más motivados, encontrar más material práctico y lograr mejoras en la capacidad analítica.

Asimismo se repasaron diversas recomendaciones para mejorar la eficacia de las visualizaciones, como que sea fácil de usar, que los cambios de estado estén bien definidos, se aporten múltiples vistas, se dé el control total al usuario para la manipulación de la visualización, e incluso para la generación de la misma, permitiéndole dar los valores de entrada que estime necesario.

Acerca de la visualización de la recursividad, se repasaron los modelos conceptuales (representaciones de un concepto con fines educativos) más importantes, tanto generales como específicos de la técnica “divide y vencerás”. Se identificaron los modelos conceptuales inductivo, metafórico, computacional, de la pila de control, de copias, de cronología de estructura de datos, o el que representa la estructura de datos en un estado concreto.

No obstante, se apreció que algunos de ellos no aparecen o lo hacen de manera muy ocasional, como es el caso del modelo de copias, el modelo de la traza de ejecución y el de metáforas mientras que otros como el árbol de activación o la vista cronológica

de la estructura de datos son empleados ampliamente junto a la explicación textual, que en muchos casos fue considerada suficiente por los autores a la hora de exponer un algoritmo.

Capítulo 3. SRec, Sistema de Animación de la Recursividad

SRec es la aplicación desarrollada en el marco de este trabajo que tiene como objetivo proporcionar visualizaciones animadas e interactivas sobre la recursividad cuya generación no requiere de esfuerzo para el usuario, proporcionando además ciertas facilidades educativas como la exportación de material gráfico o la capacidad de mostrar información detallada.

En este tercer capítulo se aborda una descripción pormenorizada de SRec. En el primer apartado se deja ver una perspectiva global de la aplicación, en el segundo se recorre la interfaz de la aplicación mientras que en el tercero se profundiza en las visualizaciones gráficas generales. A su vez, en el cuarto se abordan las vistas específicas diseñadas para la técnica “divide y vencerás”. En el quinto se explican las características de las animaciones y a su vez, en el sexto se repasan las opciones de interacción que proporciona. Por último, en el séptimo se recorre la aplicación para enumerar las distintas características, funcionalidades y capacidades.

3.1 Perspectiva Global de SRec

SRec tiene como principal objetivo asistir a profesores y alumnos durante el transcurso de asignaturas de algoritmia gracias a la generación sin esfuerzo de visualizaciones animadas y con opciones de interacción.

Para los profesores, SRec aporta la posibilidad de complementar fácilmente las exposiciones narrativas y textuales tradicionales con la visualización de múltiples ejemplos, enriqueciendo así las clases e incluso el material didáctico (apuntes, transparencias) gracias a las facilidades de exportación.

Para los alumnos, la aplicación abre la puerta a la posibilidad de generar sus propios ejemplos, repasar los ejemplos vistos en clase… para dar la oportunidad de comprender, analizar o depurar programas y asimilar mejor los contenidos de la asignatura.

Tanto profesores como alumnos pueden adoptar fácilmente la herramienta gracias a que su construcción ha seguido un enfoque de uso sin esfuerzo, minimizando

la carga de trabajo para la creación o gestión de visualizaciones, de tal forma que la aplicación se encarga de ejecutar los programas, recopilar los datos y mostrar las visualizaciones de manera autónoma y automática con tan sólo unos pocos golpes de ratón y segundos de ejecución. La Tabla 6 recoge las distintas tareas que pueden realizar los usuarios con SRec.

Profesores	**Alumnos**
- Complementar explicaciones teóricas con la animación de ejemplos - Proponer mayor variedad de ejercicios y prácticas - Crear documentación de diversos tipos: transparencias, apuntes, artículos…	- Comprender la recursividad - Analizar algoritmos - Construir programas recursivos - Depurar programas dados o construidos por ellos mismos - Crear documentación de diversos tipos: apuntes, informes…

Tabla 6. Tareas realizables con SRec por parte de profesores y alumnos

La aplicación permite crear, ver y editar clases de código Java, que contendrán programados los algoritmos recursivos que se pretenden visualizar con SRec. Una vez que la clase está cargada y es correcta, SRec duplica esa clase de forma completamente transparente para el usuario introduciendo en la copia nuevas sentencias que le permitirán obtener información de las ulteriores ejecuciones.

Cuando se cuenta con una clase Java cargada en la aplicación, se puede ejecutar el método que se desee de entre todos los que cumplen una serie de propiedades. Una vez seleccionado el método e introducidos los valores de sus parámetros de entrada, SRec ejecuta el mismo desde la clase copiada y modificada para posteriormente y de manera automática mostrar el inicio de la visualización basándose en los datos recogidos. Esta visualización cuenta con diferentes vistas complementarias, pues es sabido que la recursividad puede ser representada de múltiples maneras [24].

Con la visualización activa, el usuario puede animar la visualización (dar pasos sencillos, realizar saltos sobre subárboles de llamadas, llegar al principio o final…), interactuar con la visualización (seleccionar valores de entrada y/o salida, seleccionar métodos visibles, configuración de formato...). Estas animaciones son totalmente interactivas y tienen un marcado carácter discreto, de tal forma que el conjunto de estados disponible es contable y cada uno de ellos es significativamente diferente a los demás. La transición de un estado al siguiente se produce al activarse una nueva llamada recursiva o al obtenerse el valor de retorno de una llamada recursiva.

Es posible pedir a SRec que muestre más información sobre la ejecución o alguna subllamada concreta, así como que se seleccionen determinadas subllamadas para, por ejemplo, estudiar la redundancia de cálculos a lo largo de la ejecución del algoritmo. Además, se pueden guardar las visualizaciones abiertas para volver a cargarlas en una sesión posterior de trabajo y así poder evitarse el proceso de carga de clases y generación de visualizaciones. Esto permite crear toda una librería de ejemplos, siempre disponibles para ser usados en pocos segundos.

SRec es una herramienta de libre distribución, disponible para su descarga a través de su página Web [60].

3.1.1 Limitaciones de Uso

Existen ciertas limitaciones a la hora de utilizar SRec que conviene conocer. El lenguaje Java, las clases y métodos, la técnica “divide y vencerás” y el sistema operativo son los protagonistas de ellas. La primera limitación es sobre el lenguaje:

- SRec está orientado a la visualización de algoritmos recursivos en cursos y asignaturas de algoritmia que se imparten en el lenguaje Java bajo el paradigma procedimental. Esto implica que pueden emplearse los tipos numéricos “int”, “long”, “short”, “byte”, “float” y “double”, junto a los tipos “char”, “String” (ambos de texto) y “boolean”, que expresa los dos valores de verdad absolutos.

Se dan tres limitaciones, detalladas a continuación, sobre las clases Java que se pueden emplear y sus métodos:

- Las clases que se carguen en SRec no podrán ser abstractas, deberán implementar todos sus métodos. Las clases abstractas son una interesante característica de Java enfocada en la orientación a objetos, aspecto que resulta superfluo durante la enseñanza de la algoritmia. Todo ello no repercute de manera alguna en la cantidad de algoritmos que SRec puede visualizar, tan sólo en ciertas particularidades a la hora de escribir los métodos que contienen los algoritmos.
- La clase que se cargue en SRec deberá tener todo el código necesario para poder ejecutar el algoritmo, de tal forma que si éste cuenta con métodos (procedimientos o funciones) auxiliares, deberán estar codificados en la

misma clase y no en otra distinta. De lo contrario, no podrán encontrarse los métodos auxiliares y aparecerán errores de compilación.

- Además, se ha añadido intencionadamente una restricción más. No podrán ejecutarse métodos que no tengan ningún parámetro de entrada. Esto es así porque para poder realizar algoritmos recursivos se necesitan valores de entrada que, al no ser manejables directamente, se van reduciendo de tamaño para conseguir alcanzar casos base. Si no se cuenta con parámetros de entrada entonces no es posible implementar algoritmo recursivo alguno, por lo que la restricción evita hacer un uso anómalo de la aplicación.

Para la visualización de la técnica “divide y vencerás” se debe tener en cuenta la siguiente restricción:

- Se requiere que los métodos que contienen algoritmos diseñados bajo esta técnica cuenten entre sus parámetros con una estructura de datos (vector o matriz), que sirve para albergar los datos del problema de entrada y parámetros numéricos enteros que ayuden a delimitar la parte de la estructura con la que se trabaja en cada subllamada recursiva. Para todos los métodos Java que cumplan estos requisitos, SRec permite la activación de dos vistas adicionales que se centran en ofrecer información sobre la estructura de datos que manejan estos problemas. Se ofrece un ejemplo de cabecera de método válida en la Ilustración 12:

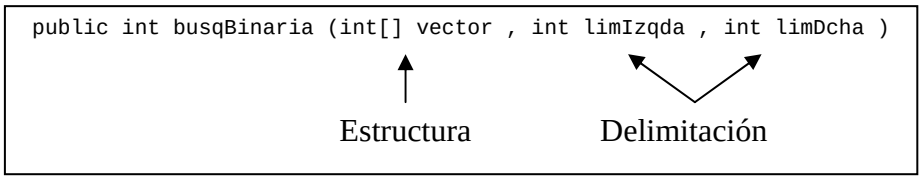


Ilustración 12. Cabecera válida de un método “divide y vencerás”

Por último, conviene contemplar una condición respecto al sistema operativo:

- Es necesario emplear SRec en un directorio del sistema de archivos en el que se posean permisos de escritura, debido a que SRec debe generar clases Java nuevas que posteriormente también tiene que compilar (generando a su vez más ficheros en disco). De no ser así, SRec no podrá compilar clases y por tanto, no podrán generarse nuevas visualizaciones, aunque mantendrá la capacidad de cargar visualizaciones guardas en formato XML en sesiones anteriores de trabajo.

3.2 Interfaz de la Aplicación

La interfaz de SRec se compone de una ventana principal, que sirve para recoger la interacción destinada a controlar el funcionamiento (carga de clases, creación de visualizaciones…) y configuración (gestión de opciones por defecto, idioma…) del programa, así como la interacción relacionada con las propias visualizaciones (manejo, control de información en cuanto a calidad y formato…) y de diversos cuadros de diálogo que tienen distintas finalidades (realizar preguntas para confirmar acciones, ofrecer información, notificar errores, permitir ajustar opciones de configuración, facilitar la ejecución de procesos de la aplicación).

La ventana de SRec consta de un total de cinco partes, tal y como se puede apreciar en la Ilustración 13 (situada en la página 83). Su función se detalla a continuación:

- Barra de menús: es similar a la de cualquier otra aplicación. Contiene cinco menús que contienen diversas funciones y opciones:
    - Archivo: permite cargar, guardar y procesar clases Java, cargar y guardar visualizaciones, exportar el contenido de las vistas a ficheros en formatos estándar (uno o varios) y cerrar totalmente la aplicación.
    - Visualización: da acceso a las opciones de configuración de las visualizaciones, permitiendo definir qué información se muestra en ellas y con qué formato.
    - Información: permite obtener más información acerca de las visualizaciones, las subllamadas representadas así como realizar el marcado y desmarcado de subllamadas.
    - Configuración: reúne las opciones de gestión de la configuración básica, así como la gestión de ficheros intermedios, la selección de idioma, la elección de la Máquina Virtual de Java, la activación de la escritura de ficheros LOG o la habilitación de la barra de herramientas de la ventana.

  - Ayuda: da acceso a un sistema de ayuda interactiva basada en hiperenlaces y a un cuadro de diálogo con la información básica de SRec y sus autores.

- Barra de herramientas: permite el rápido acceso a las opciones más empleadas de SRec, todas ellas son accesibles también desde los correspondientes menús.
- Barra de título de visualización: permite conocer el nombre del método y los parámetros de la llamada principal, de tal forma que quede totalmente claro qué algoritmo se está visualizando.
- Barra de animación: permite al usuario controlar la visualización y regular la velocidad de las animaciones. Los botones permiten:
  - Retroceder al estado inicial (activación de la llamada principal).
  - Retroceder un subárbol de llamadas (situado sobre la finalización de una subllamada, la reinicia, eliminando de la visualización a las subllamadas dependientes).
  - Retroceder un paso recursivo hacia atrás.
  - Activar animación hacia atrás (paso a paso, se muestra el transcurso de la ejecución en orden inverso hasta alcanzar el punto de inicio de la misma).
  - Pausar animación.
  - Activar animación hacia delante (paso a paso, se muestra el transcurso de la ejecución en el orden natural hasta alcanzar el punto de fin de la misma).
  - Avanzar un paso recursivo hacia delante.
  - Avanzar un subárbol de llamadas (situado al comienzo de una subllamada, muestra en un paso todas sus subllamadas dependientes y muestra el resultado de la subllamada actual).
  - Avanzar al estado final (permite ver el resultado final de la ejecución).

- Área de visualización: se compone de tres paneles, uno para contener el código Java de la clase con la que se está trabajando y dos para las vistas.
    - Panel de código: alberga el código de la clase Java que SRec tiene cargada. Permite editar el código y guardarlo. Además, alberga el código de las clases Java que sirvieron para generar una visualización XML guardada y cargada en SRec, siempre y cuando se siga encontrando disponible el correspondiente archivo Java. Contiene un panel con mensajes del compilador.
    - Paneles de vistas: son dos y pueden contener las diferentes vistas genéricas que proporciona SRec para los algoritmos recursivos y las que ofrece adicionalmente para algoritmos diseñados bajo la técnica “divide y vencerás”. En cada instante cada panel muestra una vista, tal y como se ve en la Ilustración 13. Qué vistas contiene cada panel y qué disposición ocupa cada uno (horizontal o vertical) es seleccionado por el usuario.

La Ilustración 13 recoge una captura de la ventana principal, donde se señalan las partes explicadas más importantes:

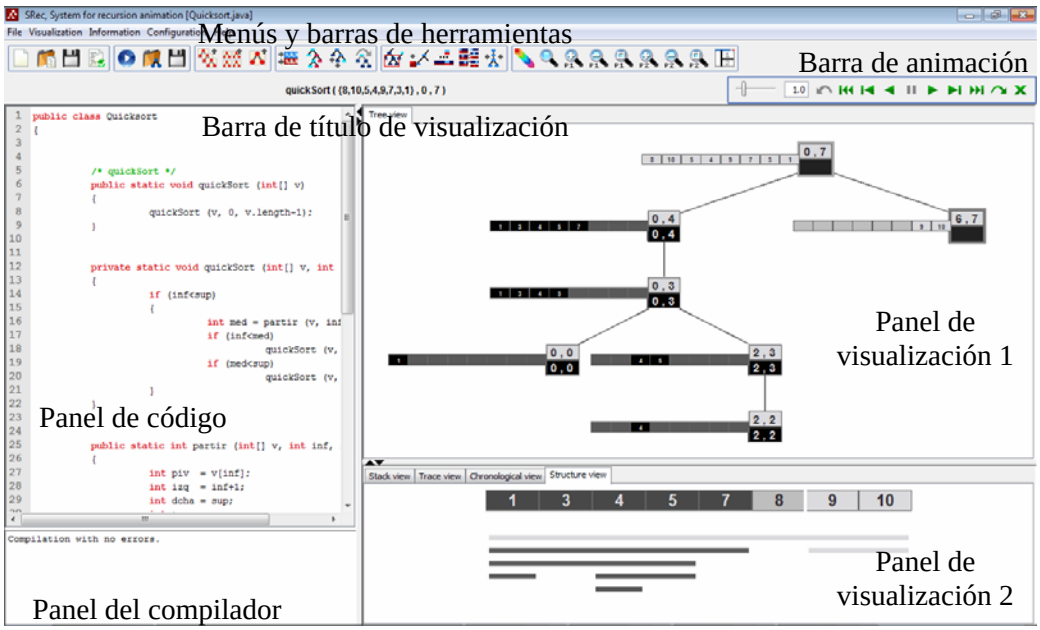


Ilustración 13. Ventana principal de SRec

3.3 Representaciones Gráficas de la Recursividad

SRec proporciona tres vistas complementarias sobre la recursividad: el árbol de activación, la pila de control y la traza de la ejecución. Todas ellas se muestran con un formato coherente y transitan de estado de manera coordinada, de tal forma que el usuario puede ir viendo información complementaria a lo largo de la animación de forma totalmente sincronizada.

Se repasan a continuación las características y funcionalidades de estas tres vistas.

3.3.1 La Vista del Árbol de Activación

El árbol de recursión es una representación jerárquica que muestra la dependencia directa entre llamadas recursivas. Cada nodo representa a una llamada recursiva y contiene el valor de sus argumentos. Las llamadas representadas a un nivel más bajo son realizadas desde la llamada a la que están conectadas pertenecientes a un nivel superior.

Un árbol de activación es, según autores como Heynes [24], una ampliación del árbol de recursión, ya que cada nodo está compuesto por dos zonas, una con el valor de los argumentos (que componen el árbol de recursión) y otra adicional con el resultado de la llamada. Se puede ver un ejemplo para el cálculo del quinto número de Fibonacci en la Ilustración 14:

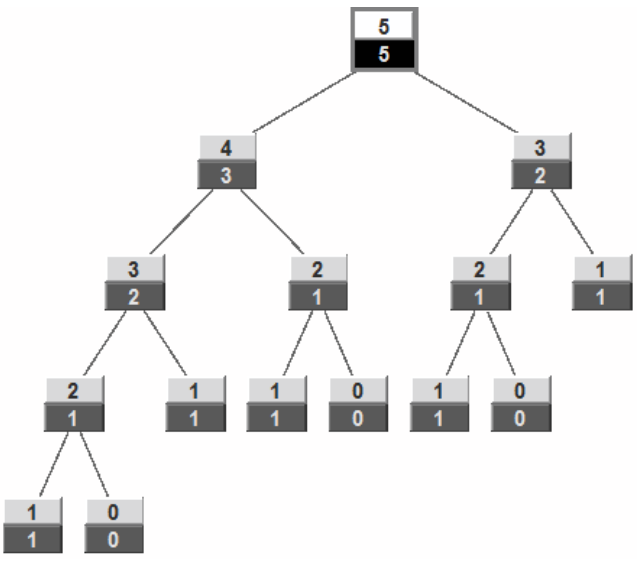


Ilustración 14. Árbol de activación generado con SRec

Una de las ventajas educativas que ofrece esta vista es que permite ver la ejecución completa, con todas las llamadas realizadas y todos los valores calculados en ellas. De esta forma, ayuda a conocer la complejidad en tiempo del algoritmo, pues es muy sencillo saber el número de llamadas realizadas y el número de veces que se realizan los cálculos.

Para esta vista, y con el fin de facilitar el manejo de árboles grandes, se facilita un visor de navegación, que aparece en la parte inferior del panel, y que muestra mediante una miniatura el árbol de activación que se está visualizando en el área principal. De esta forma, implementado la técnica global+detalle (*global+detail*) [25] el usuario puede situarse en contexto de una manera mucho más sencilla e intuitiva, y además, puede desplazarse cómodamente hasta la parte deseada. Esta vista ofrece diversas opciones de interacción a través del menú derecho del ratón.

3.3.2 La Vista de la Pila de Control

La pila de control 271[11] es un mecanismo de implementación que queda reflejado fielmente en esta vista proporcionada por SRec. Contiene los registros de activación de las llamadas pendientes en un momento dado.

SRec representa cada nodo de la pila del mismo modo que un nodo del árbol de recursión, de tal forma que cada nodo del grafo representado contiene el valor de los argumentos y el resultado, siempre que esa llamada haya finalizado su ejecución en el instante de la ejecución que se está visualizado.

Sin embargo, por sencillez, se omiten otros datos que deben estar presentes en una implementación real, como la dirección de retorno o el valor de las variables locales, pues no se ajustan a los intereses de la aplicación.

SRec visualiza la pila de control con el mismo formato que la vista del árbol de activación, lo que destaca su relación con el mismo y facilita la identificación de los mismos datos en ambas vistas. Las llamadas quedan apiladas de arriba abajo. Asimismo, es posible resaltar en el árbol de activación el camino que va desde la llamada inicial hasta la llamada actual. Este camino contiene las mismas llamadas que la pila de control. Se muestra una captura de la misma en la Ilustración 15.

Una de las ventajas educativas que ofrece esta vista es que permite investigar sobre la complejidad en espacio del algoritmo, pues muestra en cada momento de

manera simplificada el uso de memoria que se está realizando a través del número de nodos representados.

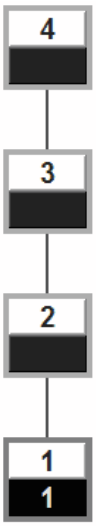


Ilustración 15. Pila de control generada con SRec

3.3.3 La Vista de la Traza de Ejecución

La vista de la traza de ejecución imprime una línea por cada entrada o salida de una llamada recursiva; en las líneas de entrada aparece el nombre y valor de los parámetros mientras que en las de salida se ofrece el valor de retorno.

```
in   quickSort: v=={8,10,5,4,9,7,3,1} , inf==0 , sup==7
  in   partir: v=={8,10,5,4,9,7,3,1} , inf==0 , sup==7
  return partir: return 5
  in   quickSort: v=={7,1,5,4,3,8,9,10} , inf==0 , sup==4
    in   partir: v=={7,1,5,4,3,8,9,10} , inf==0 , sup==4
    return partir: return 4
    in   quickSort: v=={3,1,5,4,7,8,9,10} , inf==0 , sup==3
      in   partir: v=={3,1,5,4,7,8,9,10} , inf==0 , sup==3
      return partir: return 1
      in   quickSort: v=={1,3,5,4,7,8,9,10} , inf==0 , sup==0
      return quickSort: return {1,3,5,4,7,8,9,10} , 0 , 0
      in   quickSort: v=={1,3,5,4,7,8,9,10} , inf==2 , sup==3
        in   partir: v=={1,3,5,4,7,8,9,10} , inf==2 , sup==3
        return partir: return 3
        in   quickSort: v=={1,3,4,5,7,8,9,10} , inf==2 , sup==2
        return quickSort: return {1,3,4,5,7,8,9,10} , 2 , 2
      return quickSort: return {1,3,4,5,7,8,9,10} , 2 , 3
    return quickSort: return {1,3,4,5,7,8,9,10} , 0 , 3
  return quickSort: return {1,3,4,5,7,8,9,10} , 0 , 4
  in   quickSort: v=={1,3,4,5,7,8,9,10} , inf==6 , sup==7
    in   partir: v=={1,3,4,5,7,8,9,10} , inf==6 , sup==7
    return partir: return 6
    in   quickSort: v=={1,3,4,5,7,8,9,10} , inf==7 , sup==7
    return quickSort: return {1,3,4,5,7,8,9,10} , 7 , 7
  return quickSort: return {1,3,4,5,7,8,9,10} , 6 , 7
return quickSort: return {1,3,4,5,7,8,9,10} , 0 , 7
```

Ilustración 16. Traza de ejecución generada con SRec

Cada línea se imprime con un sangrado proporcional al nivel de profundidad de la llamada recursiva. Un rastro así generado es isomorfo a un árbol de recursión, por lo

que conserva, pese a su carácter textual, la propiedad educativa de ayudar a conocer la complejidad en tiempo del algoritmo. Se muestra un ejemplo en la Ilustración 16.

Es una representación frecuente en los lenguajes funcionales, donde la recursividad es una herramienta de programación básica.

3.4 Representaciones Gráficas de la Técnica "Divide y Vencerás"

SRec admite la posibilidad, como ya se ha comentado, de aumentar hasta a cinco el número de visualizaciones simultáneas que se ofrecen sobre la ejecución de un algoritmo (si bien simultáneamente en todo momento sólo se pueden visualizar dos).

Las vistas específicas de la técnica "divide y vencerás" son especialmente interesantes en algoritmos donde se realizan labores de ordenación (por mezcla, rápida) o búsquedas (máximo, búsquedas dicotómicas, *k*-ésimo elemento), aunque también permiten una interesante visualización de algoritmos como el cálculo de la matriz traspuesta o el coloreado con formas en L de un tablero defectuoso.

Se repasan a continuación las características de estas dos vistas y la ampliación de información que contiene la vista del árbol de activación para los algoritmos de esta técnica.

3.4.1 La Vista Cronológica de la Estructura

Esta vista muestra una secuencia de estados de la estructura manejada por el algoritmo [2][71]. El orden de la secuencia se dirige desde arriba hacia abajo. Existen dos modos de visualización: el primero de ellos realza la parte que se maneja en cada subllamada frente al resto de la estructura, que permanece más oscurecido; el segundo de ellos, directamente oculta la parte no manejada, manteniendo sólo en la visualización la parte manejada.

Cada vez que una nueva subllamada recursiva es invocada, se muestra una nueva visualización de la estructura debajo de las ya existentes. Como se trata del estado de la estructura al comienzo de una subllamada recursiva, estas representaciones se muestran con el mismo color que utilizan los valores de entrada en el árbol de activación.

Cada vez que una subllamada finaliza se muestra una nueva representación de la estructura empleando el color que utilizan los valores de retorno en la vista del árbol de activación. Existen dos factores que producen variaciones en la representación de los

datos en esta vista. Por un lado, los valores de retorno pueden aparecer ligados a la entrada de la subllamada a la que pertenecen o bien puede respetarse el orden cronológico para valores de entrada (activación de llamadas) y consecución de valores de salida (finalización de la ejecución de las llamadas recursivas).

Se ofrecen en la Ilustración 17 cuatro ejemplos de esta vista para un mismo programa y estado, con diferentes configuraciones (a y b, salida no ligada a su entrada; c y d, salida ligada a su entrada; a y c, mostradas sólo partes relevantes de la estructura; b y d, estructura entera mostrada, con partes no relevantes atenuadas):

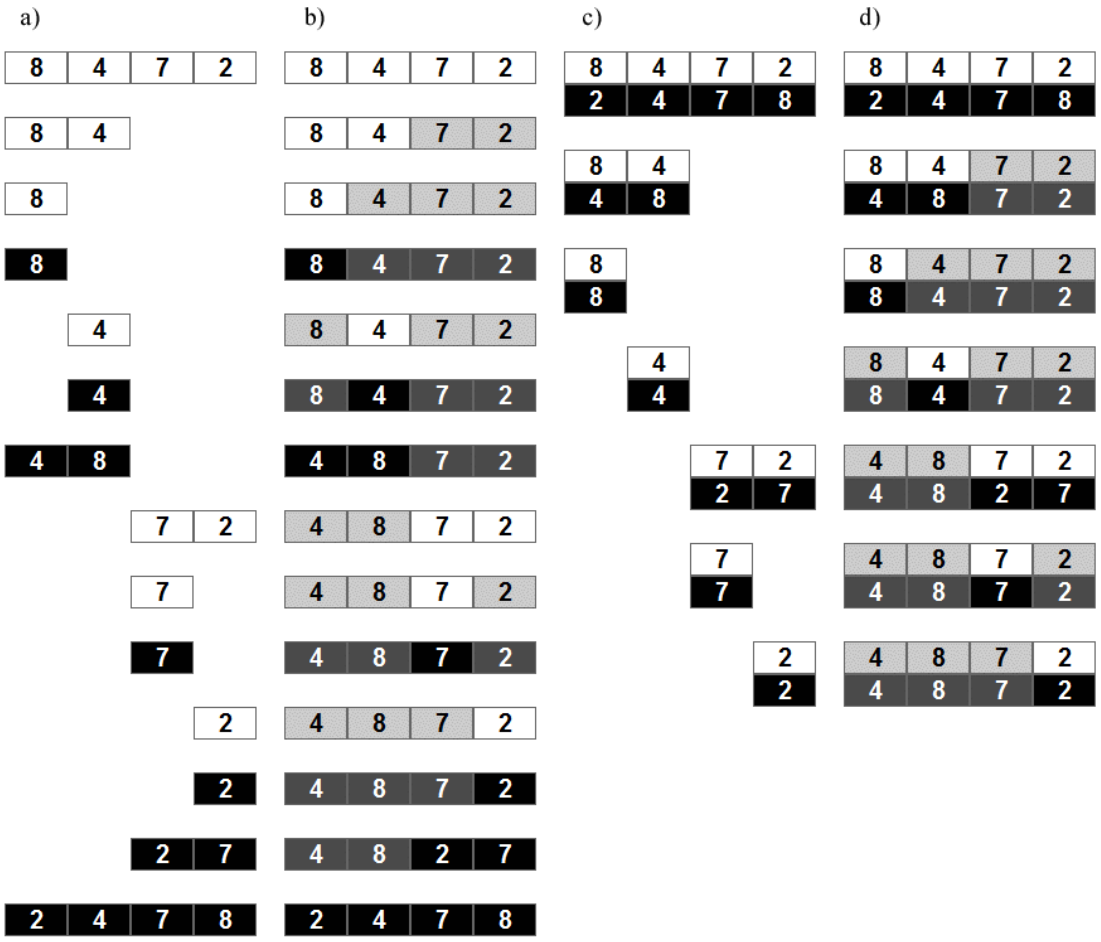


Ilustración 17. Las cuatro variantes de la vista cronológica de SRec

Con esta vista se complementa la información que ofrecen las vistas de recursividad y se facilita el seguimiento de los valores almacenados en las diferentes estructuras, lo cual puede resultar de gran interés en algoritmos de ordenación, por ejemplo. También permite verificar si cada posición de la estructura es manejada en más de una ocasión, para controlar redundancia de operaciones, o si no lo es nunca, para detectar errores en el código fuente. Además, permite representar fácil y gráficamente,

al igual que las vistas del árbol de activación y la vista de traza, la definición inductiva de los algoritmos. Se ofrece un ejemplo en la Ilustración 18 con el árbol de activación (izquierda) y la vista cronológica (derecha).

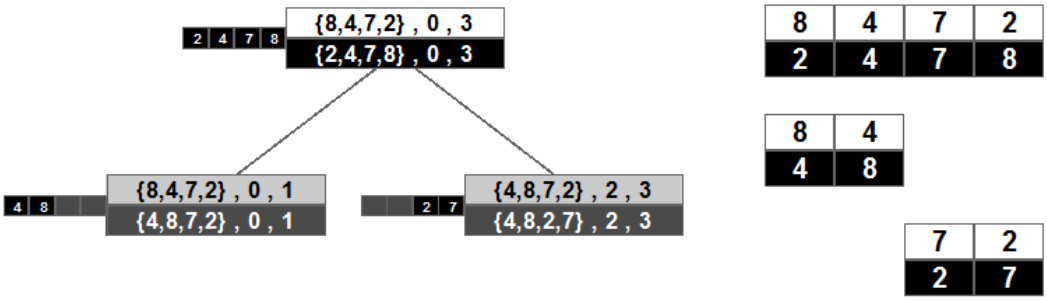


Ilustración 18. Definición inductiva con SRec

3.4.2 La Vista de la Estructura de Datos

Esta vista [75] proporciona una representación de un estado concreto de la estructura de datos (vector o matriz) manejada por el algoritmo. Ofrece una doble representación: en la parte superior se ofrece la estructura con el conjunto de valores que están coloreados según las zonas en las que se encuentran (partes ya manejadas y partes aún sin tratar). Además, se superponen líneas que reflejan la jerarquía de llamadas pendientes y la manera en que se está dividiendo la estructura de datos en cada momento.

La parte inferior de la representación deja ver qué forma adopta la jerarquía de llamadas sobre toda la estructura a lo largo de toda la ejecución. Paso a paso se van dibujando líneas (para los vectores) o cuadrados (para las matrices) que delimitan sobre qué partes se han ido aplicando llamadas recursivas. Se ofrece una representación de esta vista para el algoritmo Quicksort en la Ilustración 19:

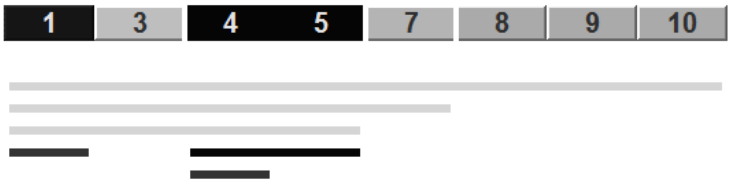


Ilustración 19. Vista de estructura de datos generada con SRec

Para las zonas aún no manejadas se emplea el mismo color que emplea el árbol de activación para los valores de entrada, al ser datos que aún no han sido manejados por el algoritmo y que por tanto, siguen estando en la entrada del algoritmo. Igualmente, las partes que ya han sido manejadas por el algoritmo muestran el mismo color que los

resultados de salida en el árbol de activación, pues ya son un producto de la ejecución del algoritmo.

3.4.3 Ampliación de la Vista del Árbol de Activación

Para los algoritmos de la técnica “divide y vencerás”, SRec, además de abrir dos vistas adicionales, aumenta la información representada en la vista del árbol de activación. Así, en cada nodo del árbol, se representa gráficamente la estructura en el lado izquierdo.

Tal representación remarca las partes que se están manejando de la estructura en la correspondiente llamada recursiva y los valores que se tienen guardados en esa parte de la estructura. La estructura además toma el color de los valores de entrada o del valor de salida en función de si se está visualizando su estado inicial (antes de la ejecución de la llamada recursiva), o su estado final (tras la realización de la misma).

El funcionamiento de la vista del árbol de activación ampliado no varía en absoluto respecto al árbol de activación convencional. En la Ilustración 20 se muestra un ejemplo de árbol de activación ampliado.

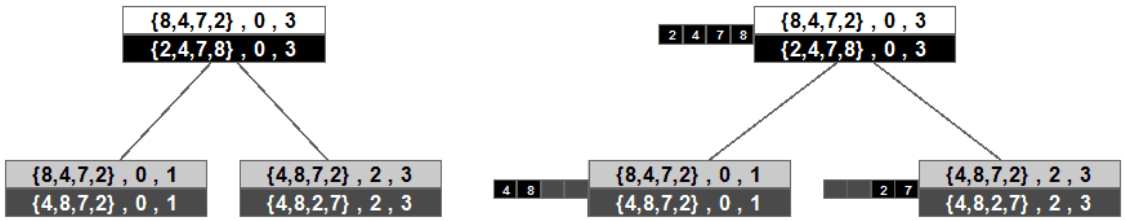


Ilustración 20. Árbol de activación normal y ampliado

3.5 Las Animaciones de SRec

Las animaciones que proporciona SRec se componen de una secuencia de estados por los que pasa la ejecución del programa. Esta secuencia nace gracias a que SRec recopila información cada vez que se entra o se sale de una subllamada acerca de los parámetros o el valor de retorno, respectivamente.

De esta forma, para cada llamada recursiva se tienen dos tipos de información:

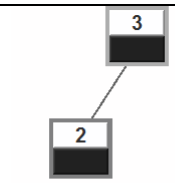
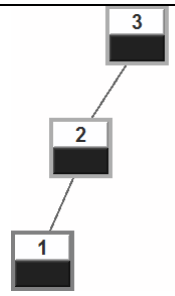
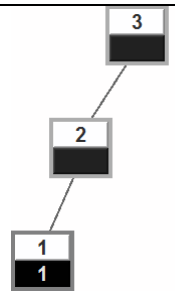
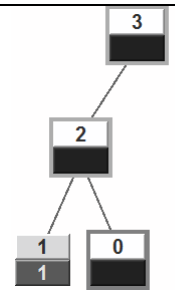
- Valores de entrada: se corresponden con los valores que tienen los parámetros de entrada al principio de la ejecución de la subllamada.

- Valor o valores de salida: se corresponde con el valor de retorno del método o bien con el valor que guardan los parámetros cuando finaliza la ejecución de la llamada recursiva si se trata de un método sin valor de retorno (tipo “void” en Java).

Cada una de las subllamadas puede encontrarse en cuatro estados a lo largo de la ejecución, que son:

- Ejecución no iniciada: no se representan en la visualización porque su ejecución aún no ha comenzado en el instante discreto de la animación que se está visualizando.
- Pendiente: su ejecución ya ha sido iniciada pero aún no ha finalizado. Aparecen representados únicamente los valores de los parámetros de entrada y, quizá, alguna o todas las subllamadas que dependen de ella, de manera parcial (pendiente) o total (finalizada), si es que tiene subllamadas dependientes (comúnmente denominadas “hijos”, siendo ésta entonces el “padre”).
- Finalizado: su ejecución finaliza justo en el instante discreto que se está visualizando, por lo que aparece su valor de retorno. La ejecución de todas sus subllamadas, si las tiene, están completamente finalizadas y se encuentran en el estado histórico.
- Histórico: su ejecución finalizó en instantes pasados, por lo que ya no interfieren en la parte restante que queda por delante de la ejecución. Son subllamadas que se pueden mostrar en colores diferentes (atenuados) o que son eliminadas de la visualización si así lo configura el usuario.

Se repasa en la Tabla 7 la ejecución natural de un pequeño subárbol de llamadas para hacer más clara la diferenciación de los distintos estados y ofrecer una noción general de cómo se transita a lo largo de una ejecución teniendo en cuenta los datos que recopila SRec. Cada línea representa un paso natural de ejecución. El ejemplo calcula el tercer número de la serie de Fibonacci.

Estado	Explicación
	Estado inicial, la llamada principal se muestra en estado pendiente. La ejecución de las subllamadas recursivas que dependen de ella (sus “hijos”) aún no ha sido iniciada, por lo que no aparecen en la visualización.
	A continuación, el siguiente paso es iniciar la primera de las subllamadas recursivas. Ambos nodos se encuentran en estado pendiente.
	El siguiente paso inicia una nueva subllamada ya que el nodo activo anterior también tiene hijos, pues no representa el alcance de un caso base. Todos los nodos se encuentran en estado pendiente.
	Como el nodo activo es un caso base, no realiza nuevas subllamadas, por lo que el siguiente paso consiste en obtener y mostrar su resultado. El nodo activo está ahora en estado finalizado, mientras que sus predecesores permanecen en estado pendiente.
	Este paso activa la subllamada del siguiente hijo del padre del nodo activo anterior, convirtiéndose además en el actual nodo activo. Su estado es pendiente, el nodo activo anterior ahora está en estado histórico y los predecesores de ambos mantienen el estado pendiente.

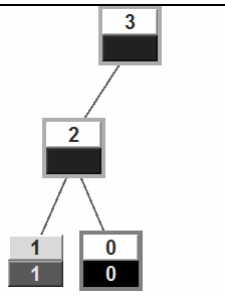
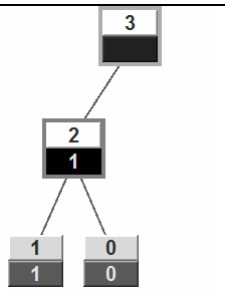
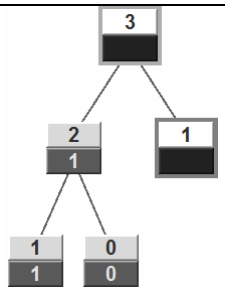
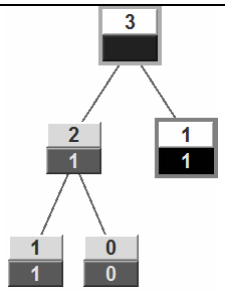
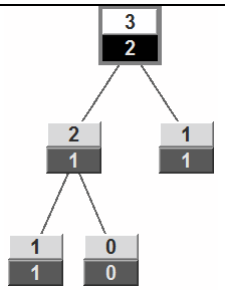
	Como el nodo activo es un caso base, no realiza nuevas subllamadas, por lo que el siguiente paso consiste en obtener y mostrar su resultado. El nodo activo está ahora en estado finalizado, mientras que sus predecesores continúan aún en estado pendiente.
	Como el nodo padre del nodo activo anterior no tiene más hijos, el siguiente paso le convierte en el nodo activo y además muestra su resultado. Sus hijos están en estado histórico y él se encuentra ya en estado finalizado. Su padre, la llamada principal, permanece en estado pendiente.
	Este paso activa la subllamada del siguiente hijo del padre del nodo activo anterior, convirtiéndose además en el actual nodo activo. Su estado es pendiente, el nodo activo anterior ahora está en estado histórico, igual que sus hijos.
	Como el nodo activo es un caso base, no realiza nuevas subllamadas, por lo que el siguiente paso consiste en obtener y mostrar su resultado. El nodo activo está ahora en estado finalizado. Se mantiene pendiente la llamada principal, mientras que los demás nodos están en estado histórico.
	Como el nodo padre del nodo activo (es decir, la llamada principal) no tiene más hijos, el siguiente paso le convierte en el nodo activo y además muestra su resultado. Todos sus hijos están en estado histórico. Es, por tanto, el estado final, pues ya se conoce el valor de la llamada principal, la que calcula el tercer número de la serie de Fibonacci.

Tabla 7. Ejecución paso a paso de un algoritmo simple.

Las visualizaciones dan la oportunidad de explorar tales estados mediante una serie de controles que se detallan a continuación. Estos controles son comunes a todas

las vistas proporcionadas por SRec y quedan accesibles en la barra de animación, ubicada en la ventana de SRec y que es representada en la Ilustración 21.

Ilustración 21. Barra de animación

- Retroceder al estado inicial (botón 2): el estado inicial es el primer momento recogido por SRec de la ejecución, el momento en que se inicia la llamada principal, con una serie de valores para sus parámetros, lo único que se puede ver en ese momento en la visualización. Por tanto, la llamada principal se encuentra en estado pendiente y el resto de subllamadas en estado no iniciado.
- Retroceder un subárbol de llamadas (botón 1): situado el nodo activo sobre una subllamada cuya ejecución está finalizada (sin ser histórica), aplicar este cambio produce que todas las subllamadas que dependen de la subllamada finalizada queden en el estado de ejecución no iniciada y la subllamada finalizada pase a estar en el estado pendiente, de tal forma que los siguientes pasos naturales en la ejecución serían volver a recorrer la ejecución de tales subllamadas para alcanzar el valor de resultado de esa subllamada activa.
- Retroceder un paso recursivo hacia atrás (botón 3): aplicado sobre una subllamada, hace que transite de estado de forma inversa a como transita de manera natural en la ejecución. Así, si la subllamada activa está en estado final, pasará a estar en estado pendiente, pasando a estar el último de sus hijos en estado final en lugar de histórico. Si la subllamada activa no tiene hijos y está en estado pendiente, pasará a estar en estado de ejecución no iniciada, y el nodo activo pasará a serlo su padre. Si la subllamada activa tiene hijos y está en estado pendiente, se revisarán recursivamente sus descendientes para aplicar este paso sobre el último de ellos que esté finalizado o pendiente.
- Animación hacia atrás (botón 4): aplica automáticamente cada cierto número de segundos graduable por el usuario pasos recursivos hacia atrás.
- Pausa (botón 5): pausa una animación automática.

- Animación hacia delante (botón 6): aplica automáticamente cada cierto número de segundos graduable por el usuario pasos recursivos hacia delante.
- Avanzar un paso recursivo (botón 7): aplicado sobre una subllamada, hace que transite de estado de la forma en que transita de manera natural en la ejecución. Así, si la subllamada activa está en estado final, pasará a estar en estado histórico, pasando a ser la llamada activa el siguiente hijo de su padre en estado pendiente o bien el propio padre en estado finalizado si no hay más nodos hijos de éste. Si la subllamada activa no tiene hijos y está en estado pendiente, pasará a estar en estado finalizado. Si la subllamada activa tiene hijos y está en estado pendiente, se revisarán recursivamente sus descendientes para aplicar este paso sobre el último de ellos que esté finalizado o pendiente.
- Avanzar un subárbol de llamadas (botón 9): situado el nodo activo sobre una subllamada cuya ejecución está pendiente, aplicar este cambio produce que todas las subllamadas que dependen de esta subllamada queden en el estado histórico y la subllamada pendiente pase a estar en el estado finalizado, de tal forma que se consigue ver de un solo golpe el subárbol de hijos junto al valor de ese resultado parcial.
- Avanzar al estado final (botón 8): el estado final es el último momento recogido por SRec de la ejecución, el momento en que se conoce el resultado de la llamada principal y, por tanto, de la ejecución completa. Por ello, la llamada principal queda en estado finalizado mientras que todas las demás subllamadas permanecen en estado histórico.

Por otra parte, la aplicación permite graduar la velocidad de las animaciones automáticas de dos formas diferentes. Por un lado, cuenta con una barra de desplazamiento que permite graduar la velocidad entre 0[1] y 5 segundos. Por otro lado, el usuario puede escribir en un cuadro de texto un número decimal que exprese el número

[1] En realidad, existe un espacio mínimo temporal de 0,3 segundos.

de segundos que desea que SRec espere entre una y otra transición de estado[2]. Ambos elementos aparecen representados igualmente en la Ilustración 21 (lado izquierdo).

3.6 Las Opciones de Interacción

Además de las animaciones, que son una herramienta muy potente para proporcionar visualizaciones de programas adecuadas y útiles, existen otras posibilidades de interacción que, en lugar de centrarse en la componente temporal como las animaciones, se centran en la espacial, aplicando cambios sobre la información relacionada con uno o varios instantes de tiempo, pero donde éste juega un papel secundario. Estas posibilidades de interacción se explican a continuación.

3.6.1 Cambiar de una Vista a otra

El usuario puede en todo momento elegir cuál de las vistas disponibles se debe visualizar. Para ello no tiene más que seleccionar la pestaña correspondiente en alguno de los dos paneles de visualización. Se muestran en detalle estas pestañas en la Ilustración 22.

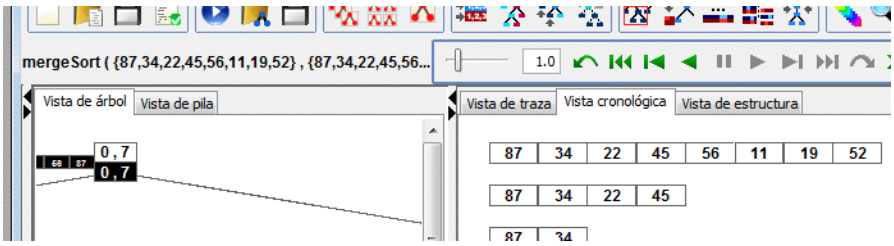


Ilustración 22. Pestañas de las vistas

3.6.2 Cambiar las Propiedades Gráficas

Mediante la opción de menú correspondiente o el botón adecuado de la barra de herramientas SRec permite el cambio de la apariencia de los elementos gracias a la graduación del color, que permite adecuar la representación a distintos tipos de escenarios (monitores, proyectores…) o a los propios gustos o necesidades del usuario. Otros factores como las distancias entre elementos o los tamaños son también

[2] Si el usuario inserta un número menor que 0,3 segundos, será éste el tiempo que transcurra entre las transiciones.

modificables. Se muestra el cuadro que permite modificar estas características en la Ilustración 23.

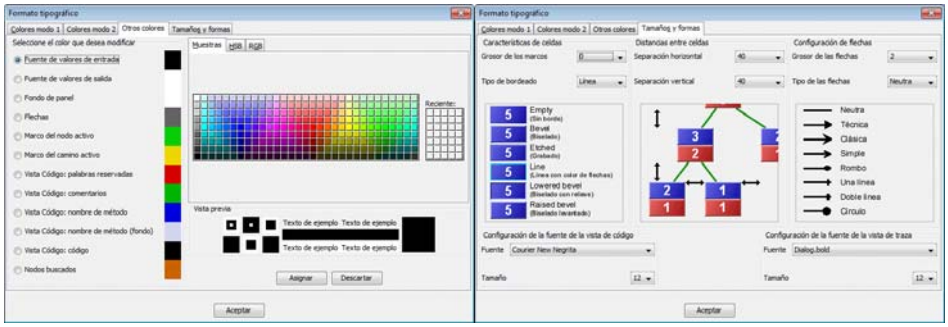


Ilustración 23. Cuadro de diálogo para variar el formato

3.6.3 Esconder y Mostrar algunos Métodos

En aquellas visualizaciones en las que están involucrados diversos métodos, SRec da la opción de eliminar o mantener los métodos que tomaron parte en la ejecución. Cada vez que el usuario decide mostrar o esconder algún método el contenido de las vistas se redistribuye para mantener la coherencia de la representación.

En la Ilustración 24 aparece un ejemplo del cuadro de diálogo que permite seleccionar qué métodos y qué parámetros de tales métodos aparecerán representados en la visualización (marcados) y cuáles deben permanecer ocultos (desmarcados). Cuando un método está desmarcado no se permite modificar su configuración pues no tendrá efecto al tratarse de un método que permanecerá oculto.

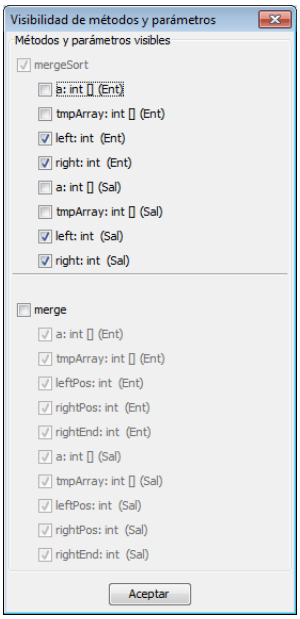


Ilustración 24. Selección de métodos y parámetros visibles

3.6.4 Esconder y Mostrar algunos Parámetros o Valores de Salida

El usuario puede elegir qué parámetros de entrada ver u ocultar y puede tomar la misma decisión respecto al valor de retorno. Si se trata de un método sin valor de retorno, entonces el usuario podrá seleccionar si desea ver el valor final de los parámetros de entrada. De esta forma, existe la posibilidad de hacer usos distintos de un mismo algoritmo.

El cuadro de diálogo mostrado en la Ilustración 24 es el empleado para seleccionar qué parámetros deben visualizarse y cuáles no.

3.6.5 Esconder y Mostrar la Entrada o Salida de las Llamadas

Yendo un poco más lejos, el usuario puede visualizar sólo valores de entrada, valores de salida o ambos en las vistas que forman las visualizaciones. De esta forma, por ejemplo, se puede transitar entre el árbol de recursión y el árbol de activación en la vista que SRec proporciona de éste último.

En la Ilustración 25 se muestra un ejemplo de árbol de activación (izquierda) y árbol de recursión (derecha), cuya diferencia es la visualización o no de los valores de salida.

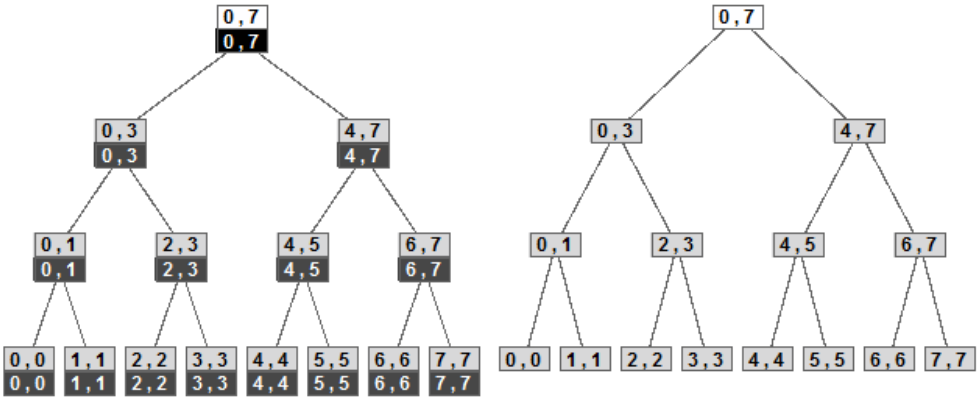


Ilustración 25. Árbol de activación y árbol de recursión

3.6.6 Esconder, Atenuar y Mostrar las Llamadas Históricas

Mediante menú y la barra de herramientas el usuario puede escoger si desea mantener, atenuar o hacer desaparecer en la imagen la representación de llamadas recursivas cuya ejecución ya haya finalizado. Se muestra un ejemplo del resultado sobre un árbol de activación en la Ilustración 26.

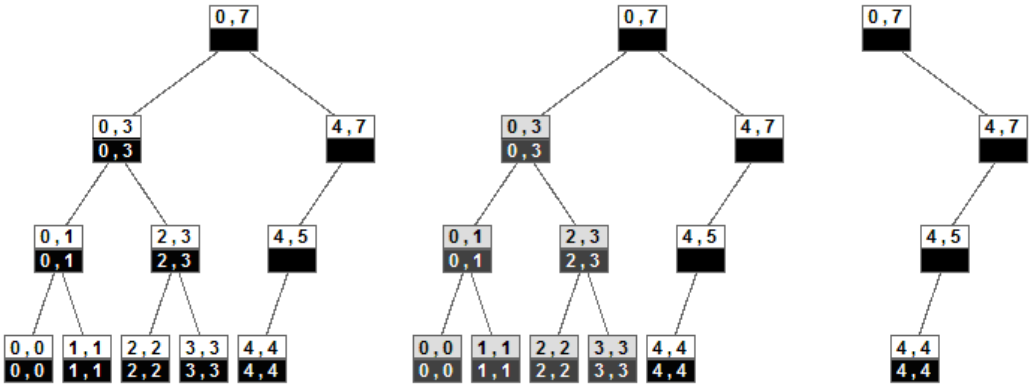


Ilustración 26. Nodos históricos mostrados, atenuados y eliminados

3.6.7 Zoom Semántico sobre una Llamada Recursiva

Mediante el botón derecho del ratón o el menú se puede acceder a cierta información sobre una llamada recursiva seleccionada. El usuario podrá saber, entre otras cosas, el estado en que se encuentra o cuántas subllamadas contiene por debajo de ella como descendientes. La Ilustración 27 ofrece un ejemplo del cuadro de diálogo que ofrece este tipo de información.

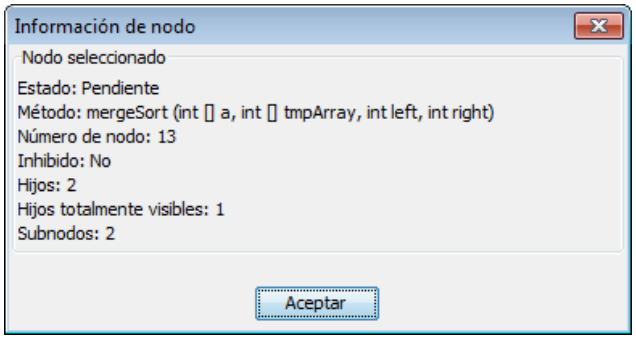


Ilustración 27. Árbol de activación y árbol de recursión

3.6.8 Reconfigurar el Layout de la Vista Cronológica de la Estructura

El usuario puede modificar la ubicación en que aparecen ciertos datos para que se respete el orden cronológico en que son utilizados los valores de entrada y obtenidos los valores de retorno o bien se mantenga cierta relación semántica entre ciertos datos, ubicando de manera próxima los valores de retorno de una llamada recursiva con los valores de entrada que los generaron.

Se ofrece un ejemplo en la Ilustración 17 (página 88), donde en los apartados (a) y (b) se muestra en estricto orden cronológico y en los apartados (c) y (d) de manera asociada.

3.6.9 Navegar en Tiempo con los Controles de Animación

SRec ofrece mediante una intuitiva barra de animación (mostrada en la Ilustración 28) el catálogo más completo de controles de animación existente entre las aplicaciones de visualización de programas con fines educativos.

Ilustración 28. Barra de animación

3.6.10 Hacer Activa o Remarcar una Llamada Recursiva

En cualquier momento el usuario puede elegir cambiar el instante de visualización que está viendo seleccionado una llamada recursiva de entre las que están visibles para pasar a visualizar el instante en que esa llamada recursiva se activa. De esta forma se proporciona un modo más de control sobre el tiempo que se visualiza dentro del conjunto finito de estados de la animación.

3.6.11 Navegar en Espacio

SRec ofrece la posibilidad de navegar en espacio mediante el zoom geométrico, la barra de desplazamiento o el visor global de navegación (se ofrece un ejemplo en la Ilustración 29). El zoom geométrico gradúa la escala de visión y, en consecuencia, la cantidad de información que se muestra en la pantalla, permitiendo que esta limitación sea direccionada.

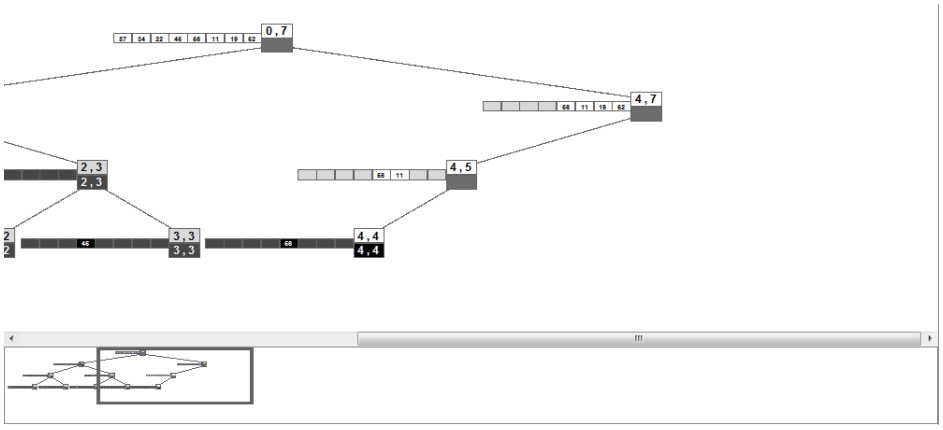


Ilustración 29. Uso de las técnicas “Global+detalle” y “Panning+scrolling”

La barra de desplazamiento permite mediante la técnica *Panning+scrolling* situarse en áreas concretas de las distintas vistas para centrarse nuevamente en cierto conjunto de datos, cercanos entre sí. Por su parte, el visor global de navegación ofrece el

contexto de lo que se ve en la vista de detalle del árbol de activación (visualización *global+detail*), dejando que el usuario sepa si está visualizando subllamadas con un alto nivel de profundidad o si se trata de llamadas ubicadas al principio (izquierda) o al final (derecha) de la ejecución.

3.6.12 Resumen de las Opciones de Interacción

Se presenta ahora un marco general de técnicas de interacción [98] para poder catalogar las diferentes posibilidades de interacción descritas anteriormente.

- Técnica Representar (Encode): Las opciones de interacción recogidas en esta técnica permiten al usuario modificar las bases de la representación visual y apariencia de los datos, como por ejemplo el color, la ubicación o el tamaño.
- Técnica Conectar (Connect): En esta técnica entran aquellas opciones de interacción que permiten resaltar asociaciones y relaciones entre elementos visibles en la visualización.
- Técnica Filtrar (Filter): Las opciones de interacción que alberga esta técnica permiten al usuario escoger el conjunto de elementos de datos mostrados basándose en algún criterio.
- Técnica Abstraer/Detallar (Abstract/Elaborate): En esta técnica se incluyen aquellas opciones de interacción que dan la posibilidad de cambiar el nivel de abstracción de los datos representados.
- Técnica Explorar (Explore): La exploración es una técnica que permite al usuario examinar diferentes subconjuntos de datos navegando entre ellos.
- Técnica Reconfigurar (Reconfigure): Esta técnica proporciona al usuario diferentes perspectivas sobre los datos permitiendo cambios en la disposición gráfica de los mismos.
- Técnica Seleccionar (Select): La posibilidad de marcar o realzar uno o varios elementos de la representación queda recogida por esta técnica con el fin de mantener fácilmente su rastro visible en la visualización.

A continuación se resumen en la Tabla 8 las posibilidades de interacción debidamente ubicadas en función de la técnica que aplican. Además, se reparten en dos

columnas, atendiendo si afectan a la utilización de la componente del espacio o del tiempo.

Técnica	Espacio	Tiempo
Representar	- Cambio de formato - Cambio de vista	
Conectar	- Enlazado de datos dependientes - Sincronización de formato entre vistas	- Sincronización del instante de visualización entre vistas.
Filtrar	- Métodos - Parámetros - Entradas/Salidas	- Llamadas históricas - Nodos saltados
Abstraer/ elaborar	- Zoom geométrico - Zoom semántico	- Información general de la visualización
Explorar	- Panning+scrolling - Vista global+detalle	- Animación
Reconfigurar	- Disposición de datos en vista cronológica de la estructura	
Seleccionar		- Seleccionar nodos

Tabla 8. Opciones de interacción clasificadas según la técnica de interacción.

3.7 Repertorio de Características y Funcionalidades

Se listan a continuación todas las funcionalidades de SRec, albergadas en los menús de la ventana de la aplicación (y las más importantes y utilizadas también en la barra de herramientas), que serán recorridos a continuación.

3.7.1 Menú "Archivo"

- Nueva clase: permite comenzar a escribir una nueva clase Java desde cero. Se necesita que el usuario aporte el nombre y la ubicación de la misma en el sistema de ficheros.
- Guardar clase: actualiza el contenido de la clase que se está editando en el sistema de ficheros.
- Cargar y procesar clase: permite escoger una clase ya existente para cargarla y procesarla, dejando disponibles sus métodos para ser ejecutados y visualizados.
- Procesar clase: se reprocesa la clase Java que ya está cargada para que entren en vigor los cambios efectuados sobre su contenido.

- Nueva animación: se insta a elegir un método de la clase previamente cargada para ejecutarlo y visualizarlo.
- Cargar animación: se permite elegir una visualización guardada previamente en el sistema de ficheros para su utilización con SRec.
- Cargar animación GIF: se permite elegir una animación generada previamente en formato GIF animado para su visualización secuencial automática con un software compatible o SRec (el formato GIF no permite interactividad).
- Guardar animación: permite guardar la visualización en uso para poder seguir utilizándola con posterioridad sin necesidad de generarla de nuevo.
- Exportar animación como GIF: permite guardar en formato GIF animado una de las vistas de la visualización en uso (excepto la vista de traza).
- Exportar capturas de animación: permite exportar una secuencia de capturas pertenecientes a cada uno de los estados que forman la animación de la visualización en uso (excepto la vista de traza).
- Exportar captura: permite exportar el contenido visible en ese instante de la vista elegida (excepto la vista de traza).
- Exportar traza: se exporta en formato HTML el contenido visible en ese instante de la vista de traza.
- Salir: cierra la aplicación.

3.7.2 Menú "Visualización"

- Datos de entrada y salida: permite elegir entre visualizar sólo parámetros de entrada, sólo valores de retorno o ambos.
- Métodos y parámetros: permite elegir, en caso de que haya más de un método involucrado en la ejecución, qué métodos se visualizan y cuáles permanecen ocultos; de igual forma se pueden seleccionar qué parámetros se desea ver de cada uno de los métodos involucrados
- Nodos históricos: permite elegir si se mantienen, atenúan o se eliminan en la visualización los nodos de llamadas recursivas cuya ejecución ya finalizó.

- Mostrar subárboles en saltos: permite elegir si se muestran o eliminan los subárboles que son saltados en un único paso.
- Mostrar visor de navegación: permite elegir si se muestra la vista global o no en la vista del árbol de activación.
- Mostrar estructura en la vista del árbol: permite elegir si se muestra en cada nodo o no el contenido de la estructura de manera gráfica (sólo para algoritmos de la técnica “divide y vencerás”).
- Mostrar estructura completa en la vista cronológica: permite elegir si se muestra la estructura íntegra o sólo las partes manejadas en cada una de las llamadas recursivas (sólo para algoritmos de la técnica “divide y vencerás”).
- Mostrar la salida ligada a la entrada en la vista cronológica: permite elegir si se muestran los valores de retorno asociados a su respectiva entrada o bien se representan todos los datos en estricto orden cronológico (sólo para algoritmos de la técnica “divide y vencerás”).
- Árboles colapsados: abre la posibilidad de limitar el espacio ocupado en la visualización del árbol de activación en función de los nodos visibles gracias a la compactación del mismo.
- Formato tipográfico: permite seleccionar los colores y propiedades gráficas de los elementos de todas las vistas, así como establecer criterios de coloreado de métodos.
- Configuración de zoom: herramienta para ofrecer con mayor o menor acercamiento el contenido de las vistas.
- Ubicación de vistas y paneles se da la opción de ubicar las vistas disponibles en alguno de los dos paneles de visualización existentes, además permite elegir cómo se distribuyen estos dos paneles en la ventana: vertical u horizontalmente.

3.7.3 Menú “Información”

- Información de la animación: se ofrece en un cuadro de diálogo información básica sobre la visualización (número de llamadas recursivas realizadas, métodos involucrados, nodos visibles, nodos ocultados, etc.)

- Información sobre el nodo activo: se ofrece en un cuadro de diálogo información sobre la llamada activa (método al que pertenece, valores totales, estado en el que se encuentra, número de subnodos, etc.)
- Búsqueda de llamadas: permite buscar basándose en el valor de los parámetros o del valor de retorno llamadas que cumplen tales criterios para resaltarlas.
- Restauración de llamadas: elimina el resaltado efectuado en búsquedas de llamadas anteriores.

3.7.4 Menú "Configuración"

- Restaurar configuración: instaura nuevamente la configuración que SRec tiene guardada como configuración por defecto.
- Abrir configuración: permite elegir una configuración guardada en el sistema de ficheros para aplicarla en SRec.
- Guardar configuración: permite guardar en el sistema de ficheros la configuración vigente para poder cargarla posteriormente.
- Guardar configuración por defecto: guarda la configuración vigente como configuración por defecto para que sea cargada al lanzarse la aplicación.
- Habilitar archivo LOG: permite elegir si se desea generar un fichero de texto plano que guarde la interacción del usuario con SRec.
- Archivos intermedios: permite elegir si se desea mantener en el sistema de ficheros algunos archivos generados por SRec durante el procesamiento de las clases Java.
- Máquina virtual: permite elegir la máquina virtual en los casos en los que SRec no pudo encontrarla automáticamente al arrancar.
- Idioma: permite elegir el idioma en que se muestra SRec entre aquellos que estén disponibles.
- Mostrar/ocultar barra de herramientas: permite mostrar u ocultar la barra de herramientas.

3.7.5 Menú "Ayuda"

- Temas de ayuda: permite abrir la ayuda interactiva basada en hipervínculos.
- Sobre SRec: cuadro de diálogo con información sobre la autoría de SRec.

3.8 Conclusiones del Capítulo

En este capítulo se ha realizado una revisión general sobre las características y funcionalidades de SRec. Así, se ha podido conocer el objetivo fundamental de la aplicación, que no es otro que ayudar a profesores y alumnos durante el transcurso de asignaturas de algoritmia gracias a la generación sin esfuerzo de visualizaciones animadas y con opciones de interacción. Los profesores pueden complementar de manera fácil y dinámica las exposiciones textuales con un catálogo de ejemplos tan amplio como quieran, mientras que los alumnos pueden practicar con los algoritmos que programen para profundizar en labores de comprensión, análisis o depuración.

La interfaz de la aplicación se basa en una ventana con varios paneles que permiten ver y editar el código del algoritmo, escrito en lenguaje Java, y trabajar con visualizaciones de múltiples vistas, que permiten visualizar paso a paso automática o dinámicamente el transcurso de la ejecución de dicho algoritmo. Tal y como se ha visto, SRec proporciona tres vistas dinámicas sobre la recursividad de carácter genérico (árbol de recursión, pila de control y traza de ejecución), y dos más específicas para los algoritmos diseñados bajo la técnica "divide y vencerás" (cronológica de estructura y de estructura de datos actual).

Las animaciones de las visualizaciones permiten diversos modos de control: paso manual, paso automático (animación), salto de subárboles, salto al extremo final, permitiendo realizar estos pasos hacia delante (sentido natural de la ejecución) o hacia atrás (sentido inverso de la misma). Las transiciones de un estado a otro provocan cambios en las subllamadas involucradas.

La interacción, que como ya se ha explicado es uno de los motores que propicia la implicación y motivación del alumnado, está presente haciendo uso de diversas técnicas que conciernen en ocasiones al espacio (exploración del estado en un instante de la ejecución) y en otras al tiempo (exploración de diferentes instantes de tiempo).

Por tanto, se ha presentado la aplicación revisando sus características y haciendo un exhaustivo repaso de todas sus funcionalidades. En el Capítulo 5, “Evaluación de Eficacia en Visualización“, SRec es comparada con otras aplicaciones desde diversos puntos de vista (opciones de interacción, modelos conceptuales representados, etc.), ahí se podrán valorar fácilmente las ventajas de la utilización de SRec frente a otras aplicaciones.

Capítulo 4. Implementación de SRec

En este capítulo se abordan algunas cuestiones técnicas relacionadas con la implementación de SRec. La arquitectura de su diseño será comentada en el primer apartado de este capítulo, mientras que el segundo explorará el flujo de procesamiento de clases. El tercer apartado revisará la utilización realizada del lenguaje de marcado XML para diversos fines.

4.1 Arquitectura de la Aplicación

SRec fue diseñado empleando una arquitectura genérica, reutilizable y ampliable. Su estructura global es completamente independiente de cuáles y cuántas técnicas de diseño de algoritmos son soportadas por la aplicación, permitiendo a su vez que las visualizaciones generadas sean flexibles, de tal forma que el usuario pueda personalizar su uso a través de la configuración de diversos aspectos como el formato o la cantidad de información mostrada.

La arquitectura que sustenta las visualizaciones se basada en el patrón arquitectónico Modelo-Vista-Controlador, que persigue separar en tres componentes diferenciados los datos que maneja la aplicación, la interfaz de usuario y la lógica de control. Las aplicaciones con interfaz gráfica suelen emplear este patrón gracias a que ofrece módulos adecuados para abarcar los componentes de tales aplicaciones.

La parte de “Modelo” contiene y manipula todos los datos que debe manejar la aplicación. La parte de “Vista” se encarga de dibujar la representación de los mismos para las vistas correspondientes y hacer llegar al usuario una representación de los datos contenidos en el “Modelo”, generalmente a través de una interfaz de usuario en forma de ventana, panel o cuadro de diálogo.

Por último, el “Controlador” se encarga de albergar la lógica de control, que se activa cuando el usuario interactúa con la aplicación o cuando el tratamiento de los datos por parte del “Modelo” requiere alguna decisión tomada por el usuario. Así, el “Controlador” recoge las interacciones del usuario, invoca las órdenes oportunas hacia el “Modelo” y envía a la “Vista” la información conveniente (datos desde el “Modelo” o información generada por él mismo).

La Ilustración 30 ofrece la representación de este patrón empleado.

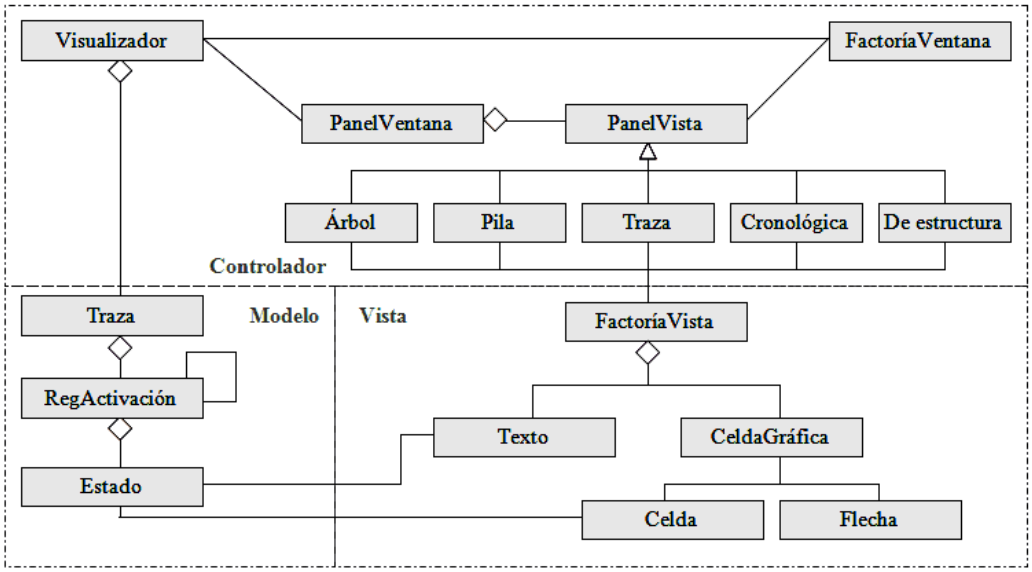


Ilustración 30. Patrón arquitectónico Modelo Vista Controlador

Las partes que varían para cada tipo de vista proporcionada quedan apartadas del resto, por lo que su modificación, eliminación o ampliación no es en ningún caso traumática para quien programe la aplicación, pues se limita a la adición, modificación o eliminación de la clase correspondiente, ubicada en la parte del "Modelo". Todo ello se traduce en una alta extensibilidad de la aplicación con muy poco esfuerzo de desarrollo.

4.1.1 Paquetes de la Aplicación

SRec ha sido desarrollado manteniendo el concepto de modularidad, ordenando el código fuente de la aplicación en varios paquetes cohesionados atendiendo a la funcionalidad y a los datos sobre los que operaba cada parte de código.

La aplicación cuenta con un total de 12 paquetes, programados ad-hoc en su mayoría, que se detallan a continuación:

- JGraph *: Este paquete contiene la librería gráfica (en su versión libre) desarrollada por la empresa inglesa homónima. La librería ofrece la posibilidad de dibujar grafos de mayor calidad que los proporcionados por Swing u otras librerías externas. Su facilidad de integración con Swing la hicieron idónea para su utilización en este programa. Esta librería ofrece distintos tipos de elementos con los que crear los grafos: celdas, flechas y puertos de conexión. Además, todos estos elementos permiten una amplia

flexibilidad en cuanto a su configuración de formato, una de las cualidades imprescindibles para poder ser utilizados en SRec.

- Java2XML *: Alberga las clases necesarias para la transformación de un archivo fuente de código Java en un documento XML. Éste es un proceso complejo, puesto que requiere analizar sintácticamente las clases y producir elementos XML que mantengan la estructura del código, sentencia por sentencia. Este módulo fue extraído desde Internet [23] y permite un uso libre del mismo siempre que se emplee sin ánimo comercial.
- Gif *: Este pequeño paquete es un proyecto libre que permite el almacenaje de animaciones GIF, una de las opciones de exportación que proporciona SRec. Está programado por varias personas independientes (entre ellas Kevin Weiner) sin ánimo de lucro, y es el resultado de años de mejoras sobre el proyecto inicial.
- Cuadros: Reúne los cuadros de diálogo de toda la aplicación, facilitando la interacción del usuario a la hora de configurar la aplicación, modificar múltiples opciones, acceder a información sobre la aplicación, gestionar los datos para las visualizaciones, mostrar barras de progreso para informar de la duración de los procesos, etc. También contiene cuadros de pregunta, y ofrece varios tipos de cuadros de error para la notificación de errores de ejecución y errores de compilación, presentados en cuadros diferentes
- Ventanas: Este paquete proporciona las dos ventanas de las que consta la aplicación. Por un lado, la ventana principal, que contiene los menús, la barra de herramientas y las visualizaciones con todas sus vistas. Ésta es la ventana que soporta la funcionalidad del programa. Por último también contiene el visor de la ayuda, que permite navegar mediante hiperenlaces y navegación secuencial por la información de ayuda que ofrece el programa.
- Paneles: Encierra en su interior varios tipos de paneles, creados cada uno de ellos con una finalidad concreta. Todos ellos están íntimamente ligados a las visualizaciones que se muestran en la ventana principal. Algunos están especializados en encapsular alguna de las vistas, el panel de edición de código, proporcionando métodos para la gestión del contenido de las vistas (actualización, adaptación...). También ofrecen información sobre los datos

que contienen (tamaño del grafo...). Otros aportan capas de funcionalidad para la gestión de las visualizaciones en la ventana o bien ayudan a configurar la disposición de las vistas o el manejo de la interfaz de las mismas.

- Eventos: Recoge ciertos eventos que pueden tener lugar durante la visualización de animaciones. Permiten que en todo momento la apariencia de la visualización sea coherente.
- Botones: Aglutina la función de creación y estandarización de los botones dentro de la aplicación. Tiene además algunos botones específicos implementados como son los botones de “Aceptar” y de “Cancelar”.
- Conf: Es el paquete que permite manejar la configuración del programa en todo momento, y su característica fundamental en cuanto a diseño es que es totalmente accesible por el resto de paquetes para facilitar el acceso a la información que contiene, que es ampliamente consultada por algunos de los demás paquetes. Se podría decir que la base de su funcionamiento es similar al que propone el patrón arquitectónico de la pizarra.
- Datos: Reúne todo el procesamiento de los datos. Así, es el encargado de gestionar la compilación de las clases, el análisis de las mismas para ver si contienen métodos visualizables, la adaptación de los métodos seleccionados por el usuario para su posterior visualización... Además, se encarga de la recuperación y almacenamiento de las trazas desde el sistema de ficheros, donde se guardan en formato XML. Por otra parte, este paquete implementa el motor lógico de las visualizaciones, lo que permite que éstas estén animadas y tengan el funcionamiento correcto y adecuado en todo momento. Es el paquete que mayor tiempo de procesamiento consume, siendo el corazón de la aplicación.
- Opciones: Este paquete gestiona el almacenamiento y recuperación de las opciones de configuración del programa, así como las configuraciones por defecto y actuales. Facilita a otros paquetes tales procesos para proporcionar una funcionalidad concreta y útil al resto del programa.
- Utilidades: Aquí se reúnen una serie de clases que contienen funcionalidades que pueden ser consideradas genéricas (pueden ser

empleadas por otras aplicaciones). Estas utilidades se han ido desarrollando según se ha ido requiriendo la funcionalidad que prestan, y en muchos casos han sido proyectadas expresamente para una implementación genérica que permita su exportación a otras aplicaciones, relacionadas o no con SRec. Herramientas adicionales para la gestión y manipulación de árboles DOM, el tratamiento y conversiones de cadenas de caracteres, la recuperación de textos en múltiples idiomas para las ventanas y cuadros de diálogo del programa, la exportación de capturas sobre paneles de la ventana... son sólo algunas de las utilidades implementadas.

- Gráfica: Contiene las clases que están orientadas a la representación gráfica de la información, aquellas que hacen uso de la librería JGraph. Son las encargadas, por tanto, de dibujar los grafos que aparecen representados en las vistas del árbol de activación y de la pila de control, implementando los algoritmos necesarios para la creación de los árboles y la ubicación adecuada de todas las celdas, tanto en el panel como en la capa de profundidad adecuada.

Los paquetes representados con un asterisco (*) son aquellos que han sido desarrollados por personas o entidades ajenas a este trabajo, de los cuales se han respetado en todo momento las licencias de distribución y utilización por las cuales son distribuidos a través de Internet. El resto de paquetes, por el contrario, han sido implementados íntegramente ad-hoc para la aplicación, en función de las especificaciones diseñadas. A continuación se presenta un diagrama de clases en la Ilustración 31 que relaciona todos los paquetes.

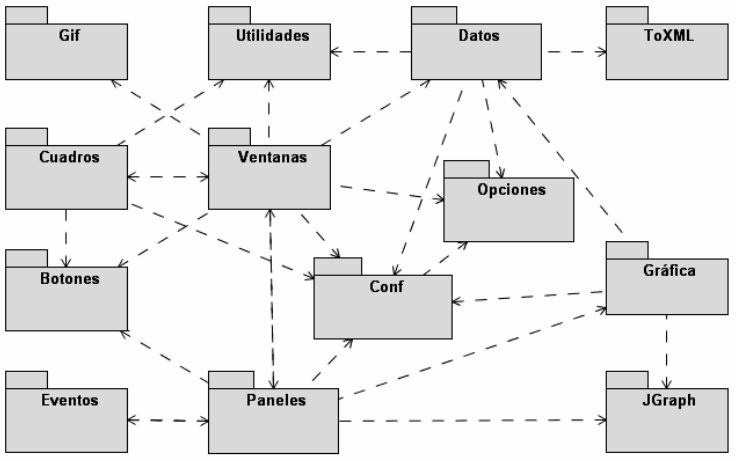


Ilustración 31. Paquetes que forman el código fuente de SRec

4.2 Flujo de Preprocesamiento de Clases Java

Una decisión clave en un sistema de animación de algoritmos es identificar qué cambios de estado durante la ejecución deben reflejarse en un cambio en la visualización del algoritmo. En el caso de SRec, tanto la activación como la finalización de la ejecución de las llamadas son los eventos que marcan un nuevo estado. No obstante, hecha esta elección, debe adoptarse una implementación concreta de entre todas las posibles.

Una opción posible es modificar el procesador del lenguaje y mantener invariado el código fuente del algoritmo que se desea visualizar. Sin embargo, en SRec se optó por la opción contraria: modificar el código fuente y mantener invariado el procesador del lenguaje. Esta opción tiene como ventajas un menor esfuerzo de implementación y la garantía de que se mantiene la corrección del procesador del lenguaje, en este caso la Máquina Virtual de Java, proporcionada pública y gratuitamente por la empresa Oracle.

Por tanto, durante el desarrollo de SRec se implementó un preprocesador que modifica el código fuente del algoritmo introduciendo sentencias en los puntos anteriormente descritos: el principio y el final de las llamadas a los métodos.

Este preprocesamiento tiene un carácter automático, de tal forma que se libera al usuario de la carga de construir animaciones, pues ni siquiera tiene por qué conocer que se realiza este proceso.

4.2.1 Etapas de Preprocesamiento

La arquitectura del preprocesamiento se muestra en la Ilustración 32 en notación de paquetes. El preprocesamiento consta de varias etapas. En primer lugar, se transforma el código Java del algoritmo en una representación equivalente en lenguaje XML, mediante el analizador Java2XML.

A continuación, la representación XML se convierte, haciendo uso de la API de DOM, en una estructura jerárquica en memoria. Posteriormente, la estructura de datos cargada se transforma, también empleando DOM, para insertar nuevos nodos en la representación jerárquica de cada método.

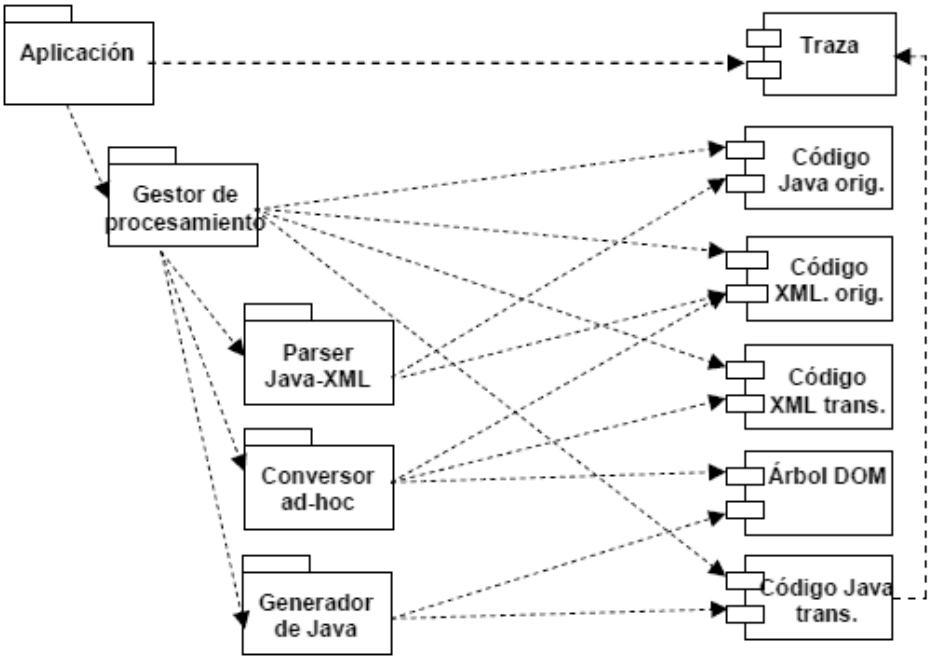


Ilustración 32. Jerarquía de paquetes del procesamiento de clases Java

Estos nodos son instrucciones que insertan en una traza información sobre el cambio de estado de la ejecución (apertura y cierre de llamadas recursivas en el caso de SRec). En un cuarto paso la estructura jerárquica manipulada se transforma de nuevo en código Java convencional, pero conteniendo ya las instrucciones necesarias insertadas en el propio algoritmo. Esta transformación desde el árbol manejado por DOM a lenguaje Java se realiza mediante un conversor específico y propio de SRec.

Finalmente SRec compila este nuevo código Java (pese a que el usuario cree que lo que se compila y ejecuta es el código que él introdujo). El fichero resultante lo ejecutará la Máquina Virtual de Java tras haber introducido el usuario sus argumentos. Como resultado de su ejecución, se crea una traza que es usada por la aplicación para crear las visualizaciones.

4.2.2 Instrucciones Generadas

Un detalle técnico son las instrucciones que se insertan en el algoritmo original escrito en Java durante esta fase de preprocesamiento. Estas instrucciones registran y almacenan los cambios de estado que deben tener un efecto en la visualización. Las instrucciones concretas y los lugares del algoritmo donde deben insertarse dependen de cada visualización particular. En el caso de SRec, se produce un cambio cada vez que se entra o se sale de una llamada. El cambio, por consiguiente, afecta a los argumentos de la misma o a su resultado.

- Las instrucciones sólo se insertan en código correcto. En caso de tener un código Java incorrecto introducido por el usuario, se muestra un mensaje informativo. Se considera que un método es correcto para SRec si, además de no levantar errores con el compilador de la Máquina Virtual de Java, se ajusta a las restricciones técnicas comentadas en el apartado “3.1.1 Limitaciones de Uso”.

```
class Fibbonaci
{
    public static int fib (int n)
    {
        int result;
        if (n<2)
            result=n;
        else
            result=fib(n-1)+fib(n-2);

        return result;
    }
}
```

```
class Fibbonaci
{
    public static int fib (int n)
    {
        Object pppppp01[]=new Object[1];
        String nnnnnn01[]={ “n” };
        pppppp01[0]=n;
        Traza.singleton().addEntrada
            (new Estado(pppppp01),”fib”,
                   nnnnnn01));

        int result;
        if (n<2)
            result=n;
        else
            result = fib(n-1)+fib(n-2);

        Object rrrrrr01[] =new Object[1];
        rrrrrr01[0] = result;
        Traza.singleton().addSalida
            (new Estado(rrrrrr01),true);
        return result;
    }
}
```

Ilustración 33. Muestra de código original y código procesado por SRec

Una vez que un código es correcto y cumple los requisitos, se procede a su procesamiento. Por ejemplo, dado un código como el de la izquierda de la Ilustración 33, que sólo incluye la formulación recursiva conocida de la serie de los números de Fibonacci, se realiza la transformación que aparece representado de la derecha en la Ilustración 33.

El primer bloque de instrucciones crea la estructura adecuada para permitir el almacenamiento en la traza del valor que tienen los argumentos en el momento en que se invoca ese método y los nombres de los parámetros, así como del método. El segundo bloque de instrucciones hace lo mismo con el resultado (o bien con los parámetros en caso de que el método no devuelva ningún valor de resultado), indicando si es un método que devuelve valor de retorno o no.

Toda esta información se almacena en una estructura lineal, implementada en la clase Traza. El método estático Singleton, que sigue el patrón de diseño homónimo, asegura que hay una sola instancia de la clase Traza durante la ejecución del algoritmo.

Cada conjunto de argumentos o cada resultado se insertan en la traza como un nuevo estado (un conjunto de valores encapsulados en un array de longitud variable).

Los nombres de variable “pppppp01” y “rrrrrr01” representan respectivamente los parámetros y el resultado. Se han elegido de forma que resulte muy improbable que un programador los declare en su algoritmo, puesto que ello provocaría colisiones de nombres y que el programa no pudiera funcionar.

4.2.3 Interacción Requerida al Usuario para Procesar Clases

Este complejo proceso, dividido en varias partes como ya se ha visto, se traduce en una interacción muy sencilla con el usuario en el caso de los algoritmos recursivos y una forma de proceder algo más compleja en el caso de algoritmos diseñados bajo la técnica de “divide y vencerás”.

Para procesar algoritmos recursivos, el usuario debe:

1. Seleccionar la opción (mediante el menú “Archivo” o la barra de herramientas) “Cargar y procesar clase”.
2. Seleccionar la clase que se desea cargar en SRec y pulsar “Aceptar”.
3. Esperar unos segundos mientras SRec carga, analiza y genera una nueva clase adecuada que será objeto de ejecución posteriormente.

Para procesar una clase que contiene al menos un algoritmo diseñado bajo la técnica “divide y vencerás”, el usuario debe:

1. Seleccionar la opción (mediante el menú “Archivo” o la barra de herramientas) “Cargar y procesar clase”.
2. Seleccionar la clase que se desea cargar en SRec y pulsar “Aceptar”.
3. Responder “Sí” a la pregunta que lanza SRec sobre si se quiere habilitar vistas específicas de la técnica “divide y vencerás” para los algoritmos diseñados bajo esa técnica.
4. Para cada método para el que se quiera habilitar estas vistas, se debe seleccionar dicho método en el cuadro de diálogo que aparece (ver Ilustración 34) e introducir en el cuadro de diálogo que aparece entonces (ver Ilustración 35) la siguiente información:

- el número ordinal del parámetro que contiene la estructura (vector o matriz).
- el número de los parámetros que contienen la delimitación del área manejada de la estructura por cada llamada recursiva (dos parámetros para vectores y cuatro para matrices).

5. Pulsar “Aceptar” para permitir a SRec finalizar el proceso de carga y procesamiento de la clase.

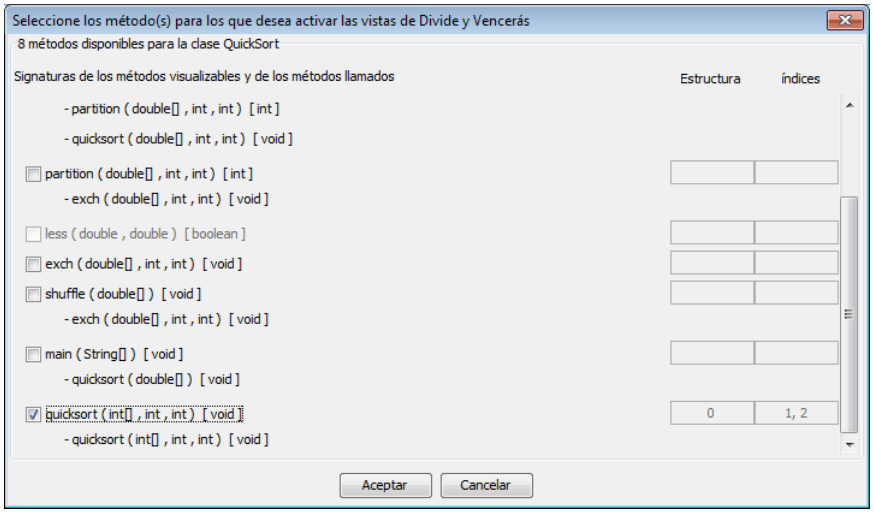


Ilustración 34. Habilitación de las vistas de “divide y vencerás”

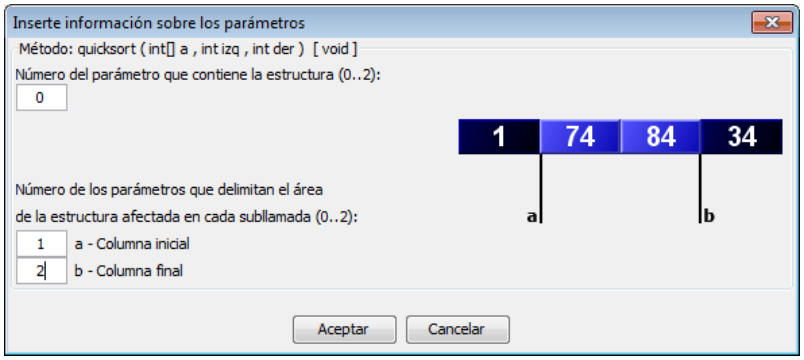


Ilustración 35. Parametrización de las vistas de “divide y vencerás”

Tras realizar este proceso, si el usuario genera una visualización de un método para el que habilitó estas vistas podrá hacer uso de un total de cinco vistas (siempre sólo dos simultáneamente) mientras que en caso contrario, serán tres las vistas que podrá emplear.

4.3 Utilización de XML

XML es un lenguaje de marcado que permite el intercambio de información estructurada entre aplicaciones, haciendo uso de ficheros y flujos de datos legibles fácilmente por el ser humano, lo que aporta un valor añadido a su utilización. La motivación por emplear este lenguaje viene por:

- XML es fácilmente usable a través de Internet, lo que puede facilitar la difusión de documentos XML generados con SRec, como por ejemplo, visualizaciones, lo que podría permitir, entre otras posibilidades, el uso online de la aplicación.
- XML es un lenguaje que puede ser manejado por una amplia variedad de aplicaciones de distinta naturaleza, plataforma y soporte, lo que puede facilitar la expansión SRec en un futuro para su utilización.
- Los documentos XML son legibles, y si se sigue la filosofía del lenguaje, deberían resultar manejables y fáciles de leer y entender para los propios humanos.
- Como consecuencia, el diseño de documentos XML debería ser sencillo y rápido. Este diseño, además, puede ser realizado de manera formal y concisa mediante lenguajes de definición de documentos.
- La escritura de programas que manejan documentos XML es, por lo general, más sencilla que la de aquellos programas que emplean formatos binarios o no estándar, pues comprobar en tales casos la corrección del formato y de los contenidos puede ser considerablemente más complejo.
- Existencia de librerías que manejan la lectura, modificación y escritura de documentos XML que permiten un fácil e intuitivo manejo de la información, evitando así invertir tiempo en la creación de librerías propias.

SRec hace uso de documentos XML en los siguientes casos: almacenamiento de los textos del programa en múltiples idiomas, almacenamiento de visualizaciones, almacenamiento de opciones de configuración, almacenamiento de parámetros para un algoritmo y representaciones de códigos Java. Los distintos documentos son explorados de manera global a continuación, si bien su especificación técnica se encuentra disponible en el Anexo 1.

4.3.1 Documento XML para la Representación de Código Java

SRec emplea el lenguaje XML para dar forma a la clase Java que el usuario selecciona para visualizar alguno de sus algoritmos. Tras esta transformación resulta mucho más sencillo el tratamiento y la manipulación del código gracias a la manipulación mediante la API de DOM, capaz de cargar documentos XML y de modificar su contenido.

Este documento XML se genera con el paquete externo Java2XML [23].

4.3.2 Documento XML para Visualizaciones

SRec da la posibilidad de almacenar en disco la visualización que está activa con el fin de que se pueda recuperar en una sesión posterior para la continuación o repetición de su uso.

El documento XML generado por SRec está orientado a satisfacer tres objetivos: almacenar la traza de ejecución completa del algoritmo que se está visualizando en el programa, guardar toda la información relevante que resulte útil a SRec para poder restaurar la visualización acerca del formato y almacenar ciertos parámetros adicionales con el fin de poder reproducir la visualización en las mismas condiciones en que fue guardada.

El almacenamiento de la traza se efectúa, como no podía ser de otra manera, mediante una definición recursiva, de tal forma que se plasma en el documento XML el árbol generado durante la ejecución del algoritmo. Es decir, existe una correspondencia directa y unívoca entre cada nodo del árbol residente en la traza y cada nodo del árbol contenido en el documento XML.

En cada nodo se almacena el nombre del método al que pertenece esa llamada (ya que se pueden registrar trazas donde se produzcan llamadas a varios métodos, de carácter recursivo o no, surgiendo así la necesidad de distinguir entre las llamadas a un método y a otro), un valor de salida, que se corresponde con el valor de retorno de esa llamada y un valor de entrada por cada parámetro que recibe la llamada de ese método.

De esta manera, se garantiza la completitud y fidelidad del proceso de guardado de la traza, lo cual podría permitir que otras aplicaciones, adaptadas al formato del documento XML pudieran cargar la traza de manera completa e incluso obviando el

resto de información contenida en el documento XML, con un fin similar (visualización) o completamente diferente.

Por otro lado, el documento XML contiene amplia información sobre el formato en que debe presentarse la traza. Esta información de formato satisface la configuración que emplea SRec en sus visualizaciones, y se corresponde con la elegida por el usuario en el momento de realizar el almacenamiento de la traza.

Respecto al formato, el documento XML contiene información sobre los colores que deben emplearse para las celdas de parámetros de entrada, celdas de valores de retorno, celdas atenuadas, flechas, fondos de panel, marcos para los nodos activos, los elementos de las vistas de traza como las líneas que representan el inicio de las subllamadas, líneas que representan el retorno de las subllamadas y los elementos de la vista de código: palabras reservadas, comentarios, nombre del método que se está visualizando, fondo de panel...

Aparte de la traza y de la información de formato, SRec guarda una serie de indicadores que le ayudan a comprender el estado exacto en el que se encontraba la animación en el momento del guardado. Esta información resulta vital para que el programa pueda restaurar la visualización, y mostrarla de manera idéntica al momento de dicho guardado.

Así, se almacena respecto a la visualización qué datos se estaban mostrando (sólo valores de entrada, sólo valores de salida, ambos), qué debe hacerse con los nodos históricos (mantenimiento, atenuación, eliminación), si deben mostrarse o no los subárboles de llamadas sobre las que se aplica un salto hacia delante, si debe aparecer o no la estructura representada en cada nodo del árbol (en algoritmos de la técnica “divide y vencerás”) y qué nivel de zoom hay que aplicar sobre las diferentes vistas.

También se almacena la dirección completa del archivo de código Java que contiene el algoritmo, con el fin de poder rescatar y mostrar el código en la vista correspondiente. Si no es posible acceder por cualquier motivo al archivo de código Java, entonces aparecerá una notificación en la vista de código.

A nivel de nodo, se almacenan varios indicadores que intentan representar el estado del mismo. En concreto, se almacena si el nodo está contenido actualmente en la pila de control, si es el nodo activo (actualmente en expansión), el estado de su entrada y salida (si son visibles o no), si es un nodo histórico, si se encuentra inhibido (es decir,

si forma parte de un subárbol que se ha desplegado mediante un salto pero que no aparece en pantalla), y si su visualización ha tenido lugar de modo completo (con todos sus nodos hijo totalmente desplegados) o no.

Por tanto, como ya se ha indicado, la información almacenada se divide en tres tipos: traza de la ejecución, formato de la visualización y parámetros de la misma, cumpliendo permitiendo recuperar la visualización íntegramente y además obtener la configuración exacta que tenía el programa cuando ésta fue guardada.

4.3.3 Documento XML para Opciones de Configuración del Programa

SRec hace uso del lenguaje XML también para almacenar sus opciones de configuración. Por un lado, guarda en este formato dos informaciones: la configuración actual y la configuración por defecto. La primera de ellas va cambiando según va el usuario modificando la configuración del programa a su gusto. La segunda es la configuración que el programa carga al arrancar cada sesión y permanece intacta mientras el usuario no decida modificarla explícitamente.

El usuario también tiene la posibilidad de guardar otras configuraciones diferentes para el programa que puede recuperar en cualquier momento, por lo que se evita así la necesidad de reconfigurar el programa para cada tipo de formato que quiera utilizar.

Todos los documentos XML que albergan en su interior la información de configuración del programa guardan el mismo formato. Por un lado, guardan la información sobre el idioma en uso, por otro guardan la selección de la máquina virtual de Java que realizó el usuario para que el programa emplee exactamente la misma versión y por otro se almacena igualmente la política de mantenimiento de archivos intermedios generados durante el procesamiento de clases que hubiese decidido el usuario.

Además se almacena cierta información que afecta directamente a las visualizaciones. En primer lugar, se define el tratamiento que debe hacer SRec sobre la información de la visualización (si debe atenuar, eliminar o mantener la información histórica en la pantalla, si se deben mostrar todos los datos de cada subllamada o sólo algunos, si se deben mostrar llamadas recursivas que han sido saltadas o no, si se debe mostrar el visor de navegación o no…).

En segundo lugar se guarda información detallada sobre el formato que tendrán las visualizaciones en todas sus vistas. Esto es así debido a que se almacena todo el repertorio de colores que se utiliza (colores para las celdas, para los fondos de paneles, para los marcos de nodos activos, para las flechas del grafo, las líneas de traza…), las formas de algunos elementos (bordes estilizados para celdas, punta de flechas…), valores de zoom y otras informaciones varias (distancias entre celdas, si se deben aplicar degradados en las celdas, qué tipo de letra se debe emplear en algunas vistas, qué grosor deben tener ciertos elementos de las visualizaciones…).

De esta forma, se puede replicar exactamente una configuración para dotar a SRec no sólo de un alto nivel de configuración y personalización, sino también de gran comodidad de manejo.

4.3.4 Documento XML para los Parámetros de un Algoritmo

Otro de los documentos XML que genera SRec está destinado a almacenar valores de parámetros de entrada para un algoritmo concreto. De esta forma, se evita la escritura repetida de ciertos valores de gran longitud que tienen intereses estratégicos para las clases magistrales. Bastará con cargar un documento XML que incluya, por ejemplo, una matriz de valores de gran tamaño para no tener que escribirlos a mano en el programa cada vez que se quiera generar una visualización.

La información que se guarda es el nombre del método, el tipo de cada parámetro y el valor deseado.

4.3.5 Documento XML para los Textos en Múltiples Idiomas

El lenguaje XML también es utilizado por SRec para el almacenamiento de todos los textos que aparecen en la interfaz de usuario.

En un único documento XML se guardan los textos de todos los idiomas, etiquetados con la información de idioma correspondiente y con un código que los identifica inequívocamente.

De esta forma, cuando SRec tiene que recuperar un texto desde el documento XML, realiza su solicitud aportando un código determinado y el idioma en que lo necesita, basándose en la configuración de idioma que ha realizado el usuario.

La estructura del documento XML le permite definir qué idiomas están disponibles en el propio documento, de tal forma que basta añadir adecuadamente el nombre de un nuevo idioma y sus correspondientes textos, para que SRec dinámicamente (sin necesidad de ser recompilado ni relanzado) pueda ofrecer la selección de un nuevo idioma. Es éste un sencillo ejemplo práctico de las ventajas que tiene la utilización de un lenguaje como XML para el almacenamiento de información estructurada y catalogada.

4.4 Conclusiones del Capítulo

Se ha revisado a lo largo de este capítulo algunos de los aspectos técnicos del interior de SRec, como su arquitectura, el flujo de procesamiento de clases y la utilización del lenguaje XML para diversos fines.

La arquitectura, genérica, reutilizable y ampliable, permite realizar fácilmente:

- Adición de nuevas vistas reutilizando los mismos elementos de representación que ya usan otras vistas o bien añadiendo elementos nuevos.
- Ampliación con nuevas técnicas de diseño, y vistas específicas para ellas.
- Modificación de las vistas existentes sin afectar a otras partes de la aplicación que no sean las relacionadas estrictamente con la visualización.

La arquitectura está basada en el patrón arquitectónico Modelo-Vista-Controlador, lo que le permite separar las partes de interfaz, gestión de datos y gestión de eventos. También se ha explorado la arquitectura de paquetes de implementación, distinguiendo entre los tres desarrollados de manera externa a este proyecto (JGraph, Java2XML, Gif) y los demás, creados en el seno de este trabajo. Todos ellos presentan una alta cohesión y un bajo acoplamiento.

De esta manera, la infraestructura de la aplicación está diseñada para poder ser ampliada y mantenida sin un esfuerzo alto de desarrollo, permitiendo una constante ampliación y adaptación a las nuevas necesidades docentes que puedan surgir. Por ello, podrían proponerse futuras ampliaciones a alumnos bajo becas de colaboración, etc. con las ventajas que ello conllevaría para el grupo de investigación (descarga de trabajo) y para el propio alumno (experiencia en mantenimiento y ampliación de un sistema software ya desarrollado).

Se ha repasado en este capítulo también el flujo de procesamiento de clases Java, que permite tomar la clase Java seleccionada por el usuario, y generar otra que contenga sentencias adicionales cuya finalidad es almacenar paulatinamente a lo largo de la ejecución de los algoritmos datos sobre las subllamadas recursivas que serán utilizados para crear las visualizaciones.

Como se ha podido constatar, se trata de un proceso complejo con múltiples etapas que se traduce, de cara al usuario, en un proceso simple, breve y en su mayor parte, transparente, manteniendo la sencillez como esencia de la aplicación, y logrando a su vez el objetivo de reducir el esfuerzo del usuario en gran medida hasta limitarlo a unos pocos clics de ratón. La aplicación también responde satisfactoriamente al requisito de ser internacionalizable gracias a su gestión de idiomas y es totalmente flexible ante el usuario, al que permite seleccionar la configuración sobre más de una decena de aspectos, así como guardar tipos de configuración para facilitar el uso diario de SRec. De esta manera, la carga de trabajo del usuario es mínima comparada con la productividad que permite desarrollar la aplicación.

Por último, se ha justificado la utilización del lenguaje XML para diversos fines, como el almacenamiento de visualizaciones interactivas, de opciones de configuración, de parámetros para el lanzamiento de métodos, o de textos de idioma. También se emplea para la representación de código Java en una fase intermedia del procesamiento de clases ya comentado. Con la utilización del lenguaje XML se consigue obtener un lenguaje de almacenamiento de datos fácilmente manipulable, que permite una sencilla depuración y ampliación de características, y que podría ser intercambiable con otras aplicaciones al ser un lenguaje de uso universal en la actualidad y fácilmente procesable.

Además, no sería costoso emplear otros lenguajes de representación vectorial como SVG, también basado en el lenguaje XML, lo que abriría las puertas a una mayor utilización y compatibilidad de SRec.

En definitiva, se han presentado los principios del diseño de diversas partes internas de SRec, justificándolo y exponiendo las ventajas que supone tal diseño.

Capítulo 5. Evaluación de Eficacia en Visualización

En este capítulo se aborda el estudio de la eficacia de SRec a la hora de visualizar algoritmos recursivos en general y diseñados bajo la técnica “divide y vencerás” en particular.

Para ello, se presenta en un primer apartado una comparativa de SRec con otras aplicaciones ya existentes acerca de cuestiones como las opciones de interacción que ofrece, la facilidad de generación de visualizaciones, los controles que aporta para manejar las animaciones o los modelos conceptuales representados. Para todas estas características se hace un pormenorizado repaso de las cualidades de las aplicaciones desarrolladas por otros autores frente a SRec. La comparación con otras aplicaciones se justifica gracias a que la bibliografía existente sí cubre todos los modelos conceptuales que proporciona SRec (incluso otros que SRec no proporciona, como el modelo de copias o el modelo de las muñecas rusas), por lo que resulta interesante desplazar el foco hacia otros sistemas software existentes.

En el segundo apartado se evalúa de SRec la capacidad ilustrativa de los modelos conceptuales pertenecientes a la técnica “divide y vencerás”. Se hace un recorrido por algoritmos y no una comparativa con otras aplicaciones ya que no existen otros programas que estén orientados específicamente a dicha técnica. Para recorrer los algoritmos se emplea la misma clasificación que se diseñó e introdujo en el apartado “2.8 Modelos Conceptuales de “Divide y Vencerás” en la Bibliografía”.

5.1 Comparación de SRec con otras Aplicaciones

Son varias las características sobre las que se pueden realizar comparaciones entre las aplicaciones existentes, mencionadas en el apartado “2.5.1 Aplicaciones Existentes para la Visualización de la Recursividad”: generalidad, esfuerzo del usuario a la hora de construir visualizaciones, características y funcionalidades de las visualizaciones generadas, posibilidades de interacción, etc.

Algunas de ellas son muy interesantes, como la generalidad, basada en los métodos que se es capaz de visualizar y el rango de tipos de datos soportado. Sobre los métodos visualizados, EROSI y SimRecur sólo visualizan un conjunto de ejemplos

predefinidos. Function Visualizer y Recursion Animator sólo visualizan el primer método escrito en el fichero de código fuente, mientras que KIEL y RainbowScheme visualizan todos los métodos recursivos. Finalmente, ETV, Flopex 2 y Jeliot son más flexibles al permitir elegir qué método se desea visualizar.

Con respecto a los rangos de tipos soportados, en la literatura se puede encontrar que EROSI y SimRecur son, de nuevo, muy restrictivos. KIEL y RainbowScheme se limitan a soportar sólo tipos primitivos y el resto de sistemas, en principio (a veces la literatura es ambigua o escueta), admiten también otros tipos más complejos. En la Tabla 9 se muestra una tabla comparativa sobre esta característica, que resume lo expuesto.

Sistema	**Selección de método**	**Tipos y datos**
EROSI	Menú prefijado	Ejemplos prefijados, valores restringidos
ETV	Automático (depende del modo de ejecución)	Cualquier tipo y valor
Flopex 2	Desarrollo visual	Cualquier tipo y valor
Function Visualizer	Automático (el primer método)	Cualquier tipo y valor
Jeliot 3	Campo de texto para escribir el nombre del método	Cualquier tipo y valor
KIEL	Automático	Tipos primitivos, con cualquier valor
RainbowScheme	Automático	Tipos primitivos, con cualquier valor
Recursion Animator	Automático (el primer método recursivo)	Sin información
SimRECUR	Menú prefijado	Ejemplos prefijados, sin información sobre valores
SRec	Menú (se permite elegir qué método(s) hacer visible)	Tipos primitivos Cualquier valor

Tabla 9. Aplicaciones en función de la generalidad.

Tal y como se puede ver, SRec ofrece la opción más flexible para el usuario, la posibilidad de elegir qué método se desea lanzar de entre todas las disponibles. No obstante, se limita la aportación de parámetros a tipos primitivos.

La segunda característica de interés para ser comparada es la cantidad de esfuerzo requerido para construir la visualización. EROSI y SimRecur sólo visualizan ejemplos predefinidos, Flopex 2 requiere que sea el usuario quien construya la visualización elemento a elemento, mientras que el resto de sistemas, entre ellos SRec, son automáticos. Es decir, generan la visualización sin intervención del usuario, lo que

supone el modo de trabajo más cómodo. En la Tabla 10 aparece la relación de esfuerzo para cada sistema.

Sistema	Modo de construcción
EROSI	Predefinido
ETV	Automático
Flopex 2	Manual
Function Visualizer	Automático
Jeliot	Automático
KIEL	Automático
RainbowScheme	Automático
Recursion Animator	Automático
SimRECUR	Predefinido
SRec	Automático

Tabla 10. Esfuerzo de construcción de las visualizaciones.

En tercer lugar, se va a establecer una comparativa tomando como referencia las características de visualización y controles de animación de diferentes sistemas, listados en la Tabla 11. Determinados modelos conceptuales son muy comunes, como por ejemplo, la traza de la ejecución, la pila de control, el árbol de recursión o el modelo de copias. Ciertos controles de animación se pueden encontrar en gran cantidad de sistemas.

Lo habitual es que los sistemas de visualización ofrezcan en total entre uno y tres modelos conceptuales, mientras que SRec aporta seis modelos, aportando además una ampliación de uno de ellos para algoritmos de la técnica “divide y vencerás”. En cuanto a controles, SRec vuelve a ofrecer el mayor catálogo de entre todas las aplicaciones, al permitir el mayor rango existente de tipos de pasos en los dos sentidos posibles.

Por otra parte, varios de los sistemas existentes no tenían opciones de interacción. Más allá de controlar la animación, el usuario no puede hacer nada con ellas ni con los elementos que son mostrados. Otros sí que dan algunas opciones, como ETV, que permite expandir o contraer nodos en el árbol de recursión, y transferir el punto de visión al comienzo de una determinada llamada a función. KIEL, por su parte, permite seleccionar un nodo con el fin de evaluar una expresión de manera completa o parcialmente automática.

Sistema	Modelos conceptuales	Controles de animación
EROSI	Modelo de copias (para variables)	Paso hacia delante Reprod. automática Salir
ETV	Modelo de copias (para código) Traza Árbol de recursión	Paso hacia delante/atrás Reprod. manual Reiniciar/Acabar Salir
Flopex 2	Pila de control Árbol de recursión	Paso hacia delante Reprod. automática Reiniciar
Function Visualizer	Modelo de copias (para código y variables)	Paso hacia delante Reprod. automática Salir
Jeliot 3	Modelo de copias Árbol de llamadas	Paso hacia delante Modo historia Reprod. manual/automática Reiniciar Salir
KIEL	Árbol de recursión	Paso hacia delante/atrás
RainbowScheme	Árbol de recursión Pila de control Código coloreado	Paso a paso Puntos de control
Recursion Animator	Modelo de copias (para variables)	Paso hacia delante/atrás Reprod. manual/automática Salir
SimRECUR	Modelo de copias (para código) Pila de control Árbol de recursión	Paso hacia delante Reprod. manual/automática Salir
SRec	Árbol de activación Árbol de recursión[3] Árbol extendido con estructura [4] Pila de control Traza de ejecución Cronológico de Estructura De estructura de datos	Paso hacia delante/atrás Reiniciar/Acabar Saltos de subárboles Reprod. manual/automática Salir

Tabla 11. Modelos conceptuales y controles de animación soportados.

Tanto KIEL como WinHIPE permiten al usuario experimentar con diferentes operaciones semánticas proporcionadas por el lenguaje de programación. WinHIPE es capaz de simplificar automáticamente la manera de mostrar las expresiones mediante la vista lógica de ojo de pez, que persigue proporcionar un compromiso entre mostrar una visión global de la expresión y la parte más relevante (es decir, la parte de la expresión

[3] Seleccionando la opción de retirar los valores de salida en el árbol de activación.

[4] Árbol de activación o de recursión con la estructura dibujada en cada nodo, sólo disponible para algoritmos diseñados bajo la técnica “divide y vencerás”

que se va a evaluar). La Tabla 12 resume las escasas opciones de interacción encontradas en los sistemas objeto de estudio.

Una vez más, las posibilidades de SRec superan ampliamente a las del resto de aplicaciones. Frente a una única opción de interacción de ETV, dos de KIEL y tres de WinHIPE, SRec proporciona un catálogo de ocho tipos de interacción diferentes sobre los datos mostrados.

Sistema	Operaciones de interacción
ETV	Expandir/Contraer nodos del árbol de recursión
KIEL	Seleccionar las operaciones semánticas del lenguaje de programación Seleccionar un vértice para evaluar su expresión
WinHIPE	Seleccionar la operación semántica del lenguaje de programación Ajuste de la simplificación automática de las expresiones usando la vista de ojo de pez
SRec	Seleccionar un nodo para hacerlo activo Seleccionar un nodo para obtener más información sobre él Seleccionar un nodo para marcar nodos iguales Mostrar u ocultar información (valores de entrada/salida, parámetros, métodos, nodos históricos, etc.) Zoom y desplazamiento Vista global y en detalle de los árboles Mostrar/ocultar subárboles Cambiar la posición relativa de ciertos datos en algunas vistas

Tabla 12. Opciones de interacción proporcionadas.

5.2 Adecuación de SRec a la Técnica "Divide y Vencerás"

SRec proporciona dos vistas específicas para los algoritmos diseñados bajo la técnica "divide y vencerás": la vista cronológica de la estructura y la vista de la estructura actual, que complementan la información que SRec muestra a través de las vistas genéricas de recursividad que ofrece: el árbol de activación (que amplía su información al visualizar algoritmos de esta técnica), la pila de control y la traza de la ejecución.

A pesar de contar con un total de cinco vistas, existen algoritmos para los cuales SRec no termina de ofrecer una visualización completamente satisfactoria.

Por ello, se ha estudiado, tomando como base la clasificación de algoritmos de esta técnica que se aportó en el Capítulo 2, la idoneidad global actual de SRec para la ilustración de la técnica para los algoritmos de cada una de las categorías de esa clasificación.

Esta clasificación, como ya se comentó en el apartado “2.6 Modelos Conceptuales de la Recursividad”, se llevó a cabo tomando como base criterios estructurales de los problemas, lo que permitió definir las siguientes categorías:

- Algoritmos que abordan problemas de carácter matemático (por ejemplo, multiplicación de matrices o cálculo del número factorial). Son algoritmos poco propensos a ser representados de manera gráfica.
- Algoritmos que hacen uso de estructuras de datos (vectores, matrices) devolviendo un valor de retorno.
- Algoritmos que hacen uso de estructuras de datos (vectores, matrices), realizando ciertas modificaciones sobre el contenido de tales estructuras y convirtiéndose éstas en el propio resultado de salida del algoritmo.
- Algoritmos con un carácter geométrico, manejando, por ejemplo, puntos en el espacio.
- Otros algoritmos que no encajan en los grupos anteriores.

5.2.1 Algoritmos Matemáticos

Los algoritmos matemáticos solucionan problemas sobre cuestiones teóricas y de carácter abstracto que provienen desde el campo de la ciencia de las matemáticas. Se reúnen en esta categoría problemas como la multiplicación de enteros de gran tamaño, la multiplicación de matrices con algoritmos avanzados, la exponenciación, el cálculo de factoriales, el cálculo de la serie de números de Fibonacci, otros más complejos como la convolución de funciones y la transformada de Fourier.

Para algunos de ellos, como la multiplicación de enteros de gran tamaño o la multiplicación de matrices SRec no proporciona una visualización detallada que realmente ayude en la comprensión, análisis o depuración del algoritmo debido a la complejidad de las tareas de división y combinación de resultados. En concreto, el problema de multiplicación de matrices no puede admitir una representación genérica fácil de diseñar ya que existen numerosos algoritmos que cumplen esta labor ofreciendo unos procedimientos diferentes cada uno de ellos.

Para otros casos, sin embargo, SRec sí es capaz de ofrecer una visualización satisfactoria, sobre todo cuando se trata de manejar un único número y realizar operaciones aritméticas sencillas. Tal y como se representa en la Ilustración 36, para los

algoritmos de exponenciación (a), la serie de Fibonacci (b) o el cálculo del factorial (c), SRec se vale de sus vistas genéricas para permitir el análisis de los mismos a los usuarios.

En estas representaciones resultan muy explícitas las operaciones matemáticas que se van realizando gracias a la variación en los parámetros de entrada de las sucesivas llamadas y en sus valores de retorno, de ahí que tanto el árbol de activación como la pila de control adquieran gran valor para estos algoritmos, evitando que sea necesario disponer de vistas adicionales.

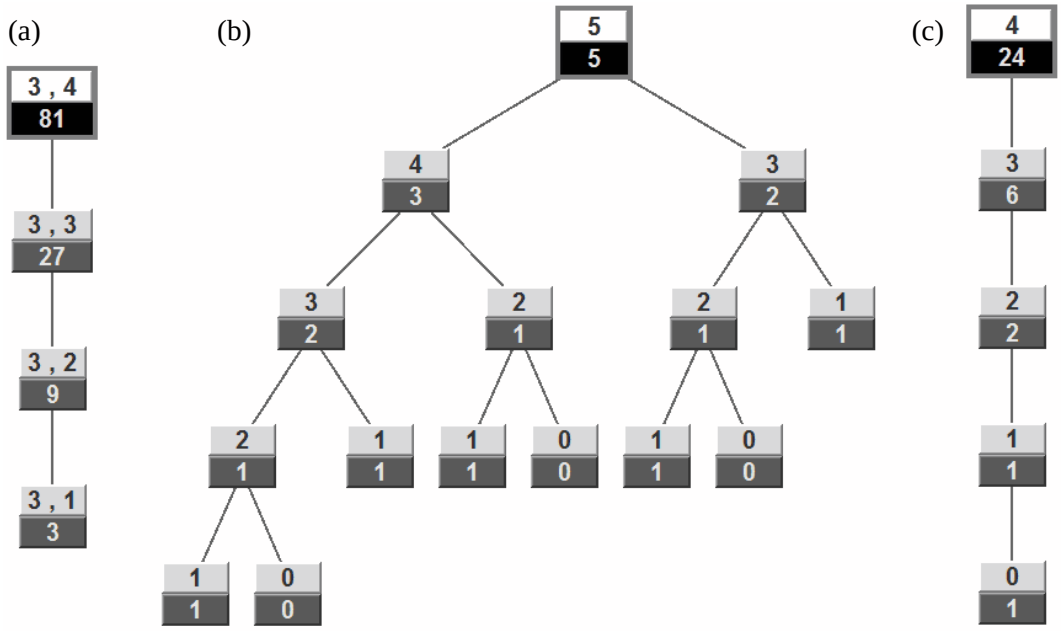


Ilustración 36. Algoritmos de exponenciación, Fibonacci y factorial

5.2.2 Algoritmos que Usan Estructuras y Devuelven un Valor

Estos algoritmos trabajan con una estructura simple de datos, como un vector o matriz, y retornan un valor simple como resultado de su ejecución, que puede ser de cualquier tipo (numérico, booleano…). En este grupo se encuentran los problemas de la búsqueda binaria, el máximo y/o mínimo de un vector, selección (búsqueda del *k*-ésimo valor más pequeño), la mediana del vector resultante de mezclar dos vectores ordenados o la búsqueda del elemento mayoritario de un vector.

SRec proporciona visualizaciones muy adecuadas para el problema de la búsqueda binaria. A través del árbol de activación SRec ilustra los valores que van tomando los índices, ayudando a ver el valor de retorno justo en el momento en que es encontrado (Ilustración 37.a), mientras que a través de su vista cronológica la selección

de las distintas partes del vector se representa de manera totalmente explícita, complementando la vista del árbol de activación. Además, la vista de estructura (Ilustración 37.b) refleja muy esquemáticamente cuántas llamadas recursivas han tenido lugar y sobre qué partes del vector han trabajado.

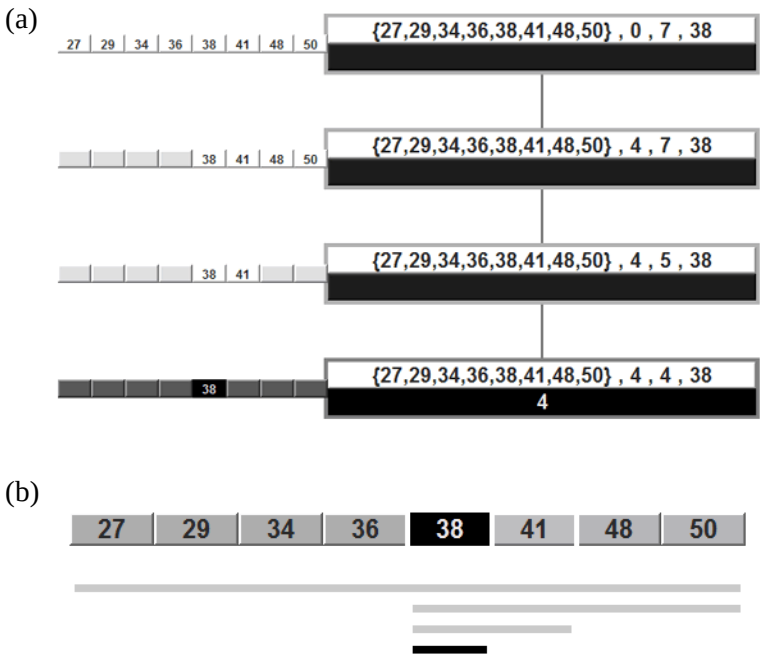


Ilustración 37. Árbol de activación y estructura para la búsqueda binaria

SRec también hace uso del árbol de activación para el caso de la búsqueda del valor máximo y de la búsqueda simultánea del valor máximo y mínimo de un vector, pues deja ver paso a paso gracias a su animación cómo se van realizando las divisiones sobre el vector y cómo se van obteniendo los diferentes valores.

SRec, por otro lado, ofrece dos vistas idóneas para hacer el seguimiento de la ejecución del algoritmo de selección, capaz de devolver el valor *k*-ésimo más pequeño de un vector no ordenado basándose en cálculos de pivote para la división del vector. En el árbol de activación (Ilustración 38) se muestran las llamadas sobre la función principal, de carácter recursivo, y sobre la función auxiliar que calcula y ubica el pivote, por lo que queda reflejado muy fielmente el proceso por el cual se va ordenando parcialmente el vector para poder determinar el valor solicitado. La vista de la estructura actual de datos permite complementar la información suministrada por el árbol centrándose en el vector y mostrando de manera gráfica la ubicación del algoritmo en una determinada parte del vector en cada subllamada que se realiza.

En la Ilustración 38 aparece representado un ejemplo ya ejecutado para el vector {17,36,41,63,29,48,10,13}, buscando el valor que quedaría en la posición 5 (empezando a contar desde cero) del mismo vector, pero ordenado.

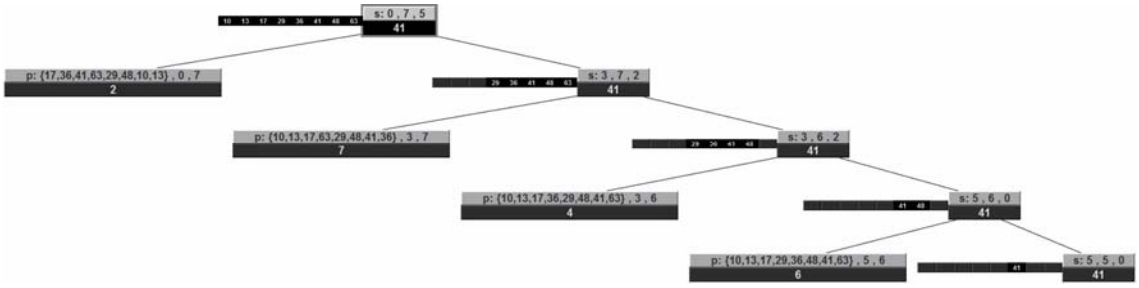


Ilustración 38. Árbol de activación para el algoritmo de selección

El árbol de activación también es propuesto para ilustrar el algoritmo que calcula la mediana del vector resultante de mezclar dos vectores ordenados. En él se puede ver qué partes de los dos vectores se van utilizando y descartando secuencialmente hasta obtener el valor que tendría la mediana. No obstante, algunas de las mejoras propuestas en el apartado “5.2.6 Propuestas para Representaciones Genéricas” serían realmente beneficiosas en este caso.

Para el algoritmo que calcula si existe un valor mayoritario en un vector, SRec proporciona, una vez más, el árbol de activación, que deja ver cómo se comporta el algoritmo y la función auxiliar de cálculo de candidato, fundamental para comprender el funcionamiento de todo el algoritmo en su conjunto.

5.2.3 Algoritmos que Usan Estructuras sin Devolver un Valor

Este grupo alberga diversos algoritmos que manipulan una estructura simple de datos, como un vector o matriz, pero sin aportar un valor de retorno, pues su ejecución se centra en la modificación del contenido de la citada estructura, cuyo estado final puede ser entendido como valor de retorno. Quedan agrupados aquí algoritmos como la ordenación de vectores Mergesort, la ordenación de vectores Quicksort el coloreado del tablero defectuoso o el intercambio de partes en un vector.

SRec ofrece para Mergesort las vistas que son más usadas en la bibliografía, con la ventaja de ofrecer la posibilidad de visualizarlo de manera animada e interactiva para cada ejemplo que desee probar el usuario. Las vistas del árbol de activación y de la sucesión cronológica satisfacen ampliamente las necesidades de profesores y alumnos, pues permiten ver de manera intuitiva cómo se obtienen las distintas ordenaciones

parciales realizadas por el algoritmo. Aparece un ejemplo de la vista cronológica en la Ilustración 39. La parte de mezcla de vectores, sin embargo, no se puede visualizar paso a paso (no es un proceso recursivo) pero no parece ser un problema, pues no suele representarse en la bibliografía en ningún caso.

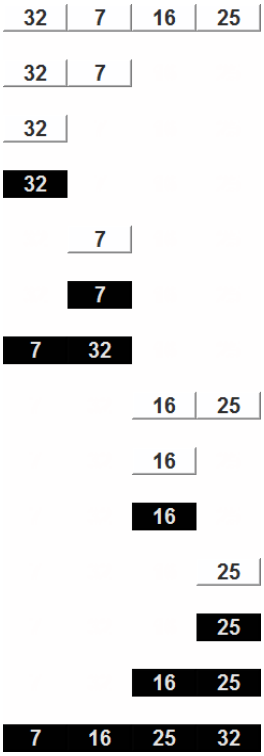


Ilustración 39. Vista cronológica para el algoritmo Mergesort

Para Quicksort SRec ofrece dos vistas interesantes: el árbol de activación y la vista de la estructura de datos actual. La primera de ellas permite ver paso a paso cómo se va dividiendo el vector, quedando explícitamente marcada la labor del proceso de cálculo del pivote, si bien éste no es mostrado paso a paso como en los libros. Se puede apreciar en la Ilustración 40.

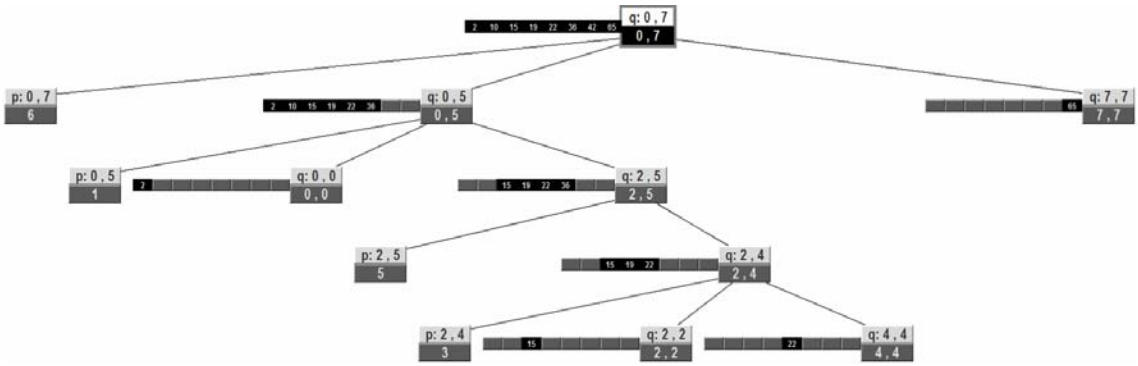


Ilustración 40. Árbol de activación para el algoritmo Quicksort

La vista de la estructura de datos, gracias a su representación esquemática de la jerarquía de llamadas, da una visión muy simple pero efectiva de qué divisiones se han realizado, dónde han ido quedando ubicados los pivotes y de cuántas llamadas recursivas han sido realizadas y sobre qué zonas del vector. Se muestra esta vista en la Ilustración 41.

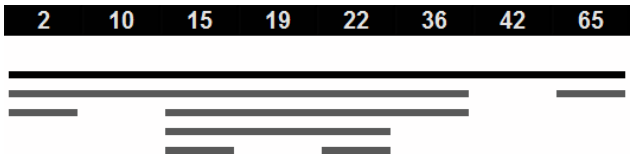


Ilustración 41. Vista de estructura para el algoritmo Quicksort

Al igual que ocurre con el algoritmo Quicksort, el problema de colorear un tablero defectuoso con figuras en forma de L concentra en la tarea de división la mayor parte de sus operaciones. SRec permite ofrecer mediante sus vistas cronológica de estructura y de estructura de datos actual dos representaciones muy precisas que permiten visualizar con gran detalle los procesos de división y coloreado, resultando muy fácil el seguimiento del algoritmo. Se ofrece una representación de la vista de la estructura de datos actual en la Ilustración 42.

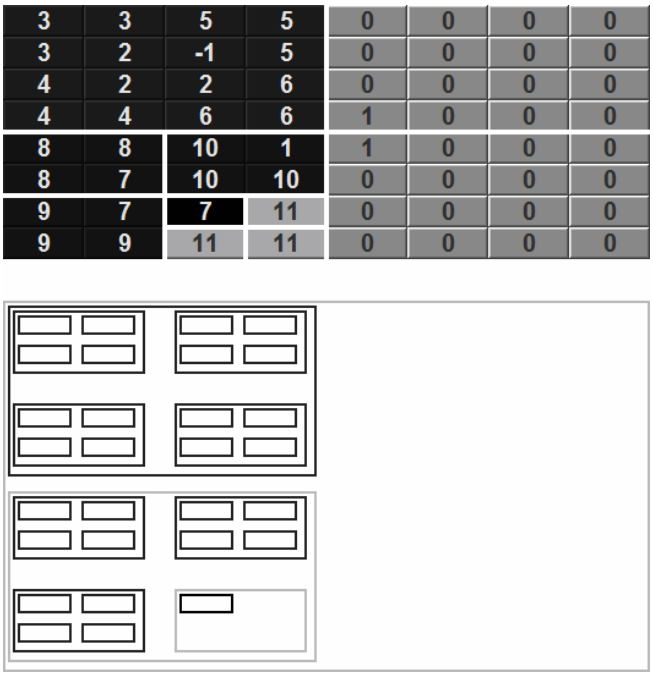


Ilustración 42. Vista estructura para el algoritmo del tablero defectuoso

El algoritmo de desplazamiento de los contenidos de un vector es sencillo de comprender y su concepto suele ser fácil de representar. SRec, en función de la semántica de los parámetros puede representar mejor o peor a través de las vistas de la

estructura de datos actual y la vista cronológica de la estructura los distintos estados por los que va pasando el vector, pero en todos los casos la vista del árbol permite hacer un seguimiento ampliamente satisfactorio de las transiciones que se producen en la ordenación de los datos.

5.2.4 Algoritmos Geométricos

Esta categoría reúne algoritmos cuyos elementos tienen un carácter geométrico, como por ejemplo, planos, puntos o líneas. Algunos de los algoritmos recogidos en esta categoría son el cálculo del par de puntos más cercanos entre sí dado un conjunto de puntos en el plano, el cálculo del número de puntos que domina cada punto del plano dado, el cálculo de polígonos convexos y el cálculo de diagramas de Voronoi.

En general, estos algoritmos son representados muy fácilmente en los libros [1][2][9][32][35][40][71] haciendo uso de ejemplos donde se sitúan puntos u otros elementos en un eje de coordenadas de dos dimensiones. A estas representaciones se añaden zonas coloreadas, remarcadas, separaciones o bien flechas que sirven para indicar cálculos, relaciones, divisiones del problema, etc.

Para estos problemas, en los que predominan las representaciones bibliográficas basadas en el dominio, las representaciones de SRec resultan muy poco expresivas debido a su carácter genérico y, habitualmente, no aportan ningún valor para ayudar en el análisis o comprensión del algoritmo que se emplea para resolver este tipo de problemas.

5.2.5 Resumen de las Representaciones Utilizadas por SRec

SRec ofrece varias vistas que se ajustan en mayor o menor grado a algunos tipos de problemas. Se refleja en la Tabla 13 para cada problema qué vistas dan un buen resultado (expresado con “X” en la tabla), qué vistas sirven como complemento pero por sí mismas dan un resultado incompleto (“x”) y qué vistas pueden usarse para hacer un seguimiento de la ejecución de los algoritmos pero sin ofrecer una contribución de gran interés ilustrativo (“~”).

Como se puede ver, existen cinco algoritmos para los que las vistas proporcionadas por SRec tienen un valor limitado. Ello da fe de que deben encontrarse nuevas visualizaciones que ayuden a completar la funcionalidad de SRec.

Problemas \ Vistas de SRec	**Árbol**	**Árbol con muestreo de la estructura en nodos**	**Pila**	**Cronológica**	**De estructura**	**Imposible visualización satisfactoria**
Multiplicación enteros grandes	~					
Multiplicación de matrices	~					
Exponenciación	X		X			
Factorial	X		X			
Serie números de Fibonacci	X		X			
Convolución y Transformada Fourier	~					
Búsqueda binaria		X		X	x	
Máximo de un vector		X		X	x	
Máximo y mínimo de un vector		X		X	x	
Selección		X		X	X	
Mediana de un vector (selección)		X		X	X	
Mediana de vector mezcla de dos vectores	~					
Elemento mayoritario de vector		X		X	x	
Mergesort		x		X	X	
Quicksort		x		X	X	
Coloreado tablero defectuoso				X	X	
Intercambio de partes en un vector	~					
Par de puntos más cercanos en plano						X
Puntos dominados en un plano						X
Polígonos convexos / Diagramas Voronoi						X
Campeonato						X

Tabla 13. Problemas para los que SRec ofrece vistas ilustrativas.

5.2.6 Propuestas para Representaciones Genéricas

Tal y como se ha explicado, hay ciertos problemas para los que no se cuenta con una representación gráfica satisfactoria mediante la utilización de SRec (tampoco en la bibliografía consultada en el apartado “2.8 Modelos Conceptuales de “Divide y Vencerás” en la Bibliografía”.

Vistas las carencias existentes en la representación de algoritmos, se aportan algunas representaciones ideadas para tales algoritmos que intentan ser a su vez lo más genéricas posible para permitir su expansión con otros algoritmos. Estas

representaciones están enfocadas a ser integradas en un sistema interactivo de visualización de programas como SRec.

Uno de estos problemas es el cálculo de la multiplicación de números grandes. Para este problema se aporta una propuesta en forma de árbol de activación ampliado (Ilustración 43) que intenta ser genérica y, por tanto, aplicable a más algoritmos mediante el encuentro de una fórmula que permita la representación de operaciones aritméticas basada en ciertas convenciones aplicables en múltiples casos.

Lo que intenta expresar la propuesta que se muestra en la Ilustración 43.a para este algoritmo es la división de los cálculos en tres partes, cada una de ellas multiplicada por la base óptima de exponenciación elevada a 2*s, s y 0 (siendo s el número de cifras que tiene cada parte de los números tras su división en dos mitades). La propuesta de la Ilustración 43.b expresa un algoritmo existente mejorado, de menor complejidad, que también suele ser enseñado en las aulas, en donde se realizan cálculos previos auxiliares sobre tres variables y que posteriormente son utilizadas para ahorrar en el número de multiplicaciones realizadas, que pasan de ser cuatro a tres.

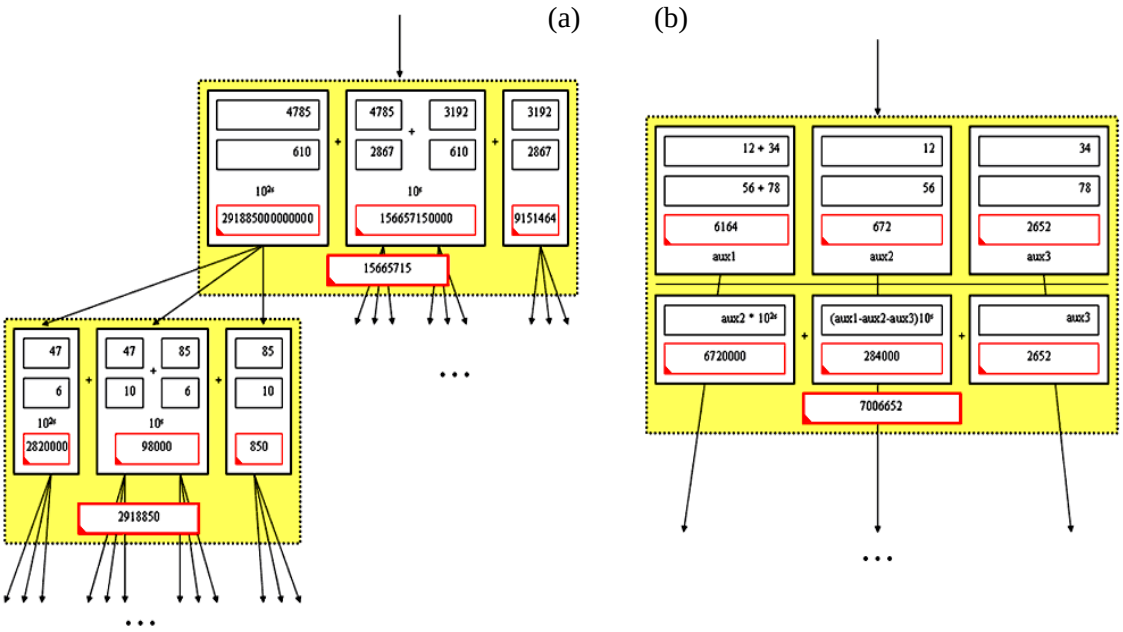


Ilustración 43. Propuesta de visualización de árbol de activación

En ambos casos los árboles se producen por las llamadas recursivas que se lanzan en ambas variantes del algoritmo. En la Ilustración 43 se muestra el ejemplo para los números 47.853.192 y 6.102.867 (a) y 1.234 y 5.678 (b).

Las flechas inferiores apuntan a nuevas llamadas recursivas, representadas por las celdas sombreadas y divididas en varias partes en función de los cálculos que se realizan. Las multiplicaciones se expresan mediante verticalidad, quedando representado cada operando por una caja, mientras que las sumas se representan de manera horizontal con el operador escrito de manera explícita. Los resultados se expresan en la parte inferior, justo debajo de los operandos, con el borde de la caja con un ángulo remarcado.

Esta representación podría extenderse a algoritmos similares en los que las operaciones matemáticas se mantienen presentes de manera importante, estableciendo así un convenio general de representación de las diferentes operaciones aritméticas básicas existentes.

Por otro lado, se proponen representaciones genéricas para algoritmos que manejan estructuras tales como vectores y matrices. Estas representaciones, basadas en el árbol de activación clásico, permiten mostrar el contenido de una o varias estructuras de datos, tanto como parámetros de entrada como valores de salida. Cada representación delimita con colores qué partes se están manejando en cada subllamada recursiva, esta delimitación se realiza mediante parámetros, que pueden estar representados en la vista o no.

Pueden ser utilizadas en algoritmos tales como la multiplicación de matrices (sin entrar en detalles específicos de la implementación, pero sí dejando ver los resultados parciales que se van obteniendo) las que aparecen en la Ilustración 44.c. No obstante, la multiplicación es sólo una de las múltiples operaciones posibles que se pueden realizar tomando como origen dos matrices, esta representación podría ajustarse a algoritmos que hagan uso de cualquiera de tales operaciones.

También se aportan en la Ilustración 44 algunas posibles representaciones para algoritmos que manejan dos vectores y que devuelven un valor de retorno, como es el caso del cálculo de la mediana del vector resultante de mezclar dos vectores ordenados. Para este tipo de algoritmos sería totalmente indicada la representación de la Ilustración 44.b, teniendo su homóloga con matrices en la Ilustración 44.d.

La representación Ilustración 44.a podría ser aplicable, por ejemplo, a un algoritmo como el de mezcla de dos vectores que utiliza el algoritmo Mergesort, al emplear dos vectores como valores de entrada y devolver uno nuevo de salida.

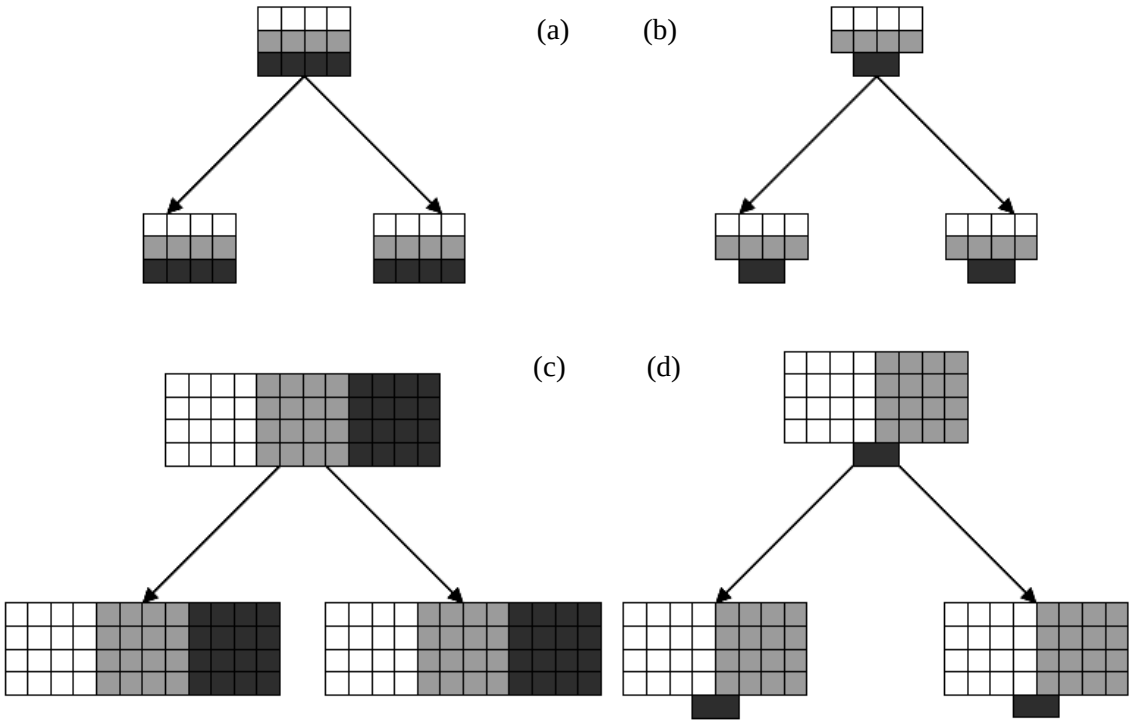


Ilustración 44. Propuestas para representación de árboles con estructuras

El grupo de algoritmos que manejan una o dos estructuras y que devuelven un valor simple obtendrían una mejora en la representación de sus ejecuciones a través de SRec si la vista cronológica añadiera en un lateral la información sobre el valor de retorno de manera asociada a cada solución parcial obtenida al finalizar cada llamada recursiva del algoritmo. De esta forma, la vista conseguiría ofrecer de manera completa la información sobre cada estado de la ejecución, como refleja la Ilustración 45.

Todas estas representaciones propuestas son, en mayor o menor grado, variaciones o ampliaciones de las visualizaciones que actualmente ya produce SRec. Suponen aún un campo no explorado totalmente y se deja su continuación como trabajo futuro para una catalogación más exhaustiva, en este momento sólo se busca establecer las bases de las propuestas, contextualizadas en una búsqueda bibliográfica que ha permitido identificar una serie de carencias y limitaciones que se pretenden superar.

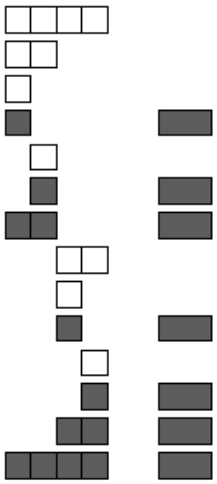


Ilustración 45. Propuesta para ampliar la vista cronológica

5.3 Conclusiones del Capítulo

En definitiva, se ha explorado la eficacia de la aplicación desde un punto de vista comparativo con otras aplicaciones y de manera analítica acerca de la técnica “divide y vencerás”.

En primer lugar, SRec ha sido comparado con otros programas acerca de diversas cuestiones. Así, se pudo ver que SRec es de los pocas aplicaciones que ofrece un menú al usuario para que este pueda elegir qué método visualizar de los que él ha cargado, permitiendo cualquier valor, siempre que éste pertenezca a un tipo primitivo del lenguaje. Otros programas dan opciones más restringidas o permiten sólo la visualización de un limitado conjunto de ejemplos prediseñados. SRec forma parte del grupo mayoritario de aplicaciones capaz de generar las visualizaciones de manera automática, sin intervención del usuario. SRec también fue comparado respecto al conjunto de modelos conceptuales que ofrece, alcanzando un total de siete, frente a los tres que suman algunas de las aplicaciones estudiadas. Esto supone una capacidad expresiva sensiblemente superior a la del resto de aplicaciones.

SRec también es la más rica a la hora de proveer controles de animación. Ninguna otra permite tantas variantes de controles en doble sentido (hacia delante y hacia atrás) como SRec, lo que le otorga una flexibilidad de manejo más amplia que la del resto de aplicaciones.

SRec también obtiene el primer lugar si se trata de comparar las opciones de interacción sobre las visualizaciones. La selección de elementos, la configuración de la

cantidad de información que se muestra y las posibilidades de navegación por algunas de las vistas consiguen hacer de SRec la aplicación con el catálogo más amplio de opciones de interacción de entre las orientadas a la visualización de la recursividad. De esta forma, se puede concluir que SRec realiza aportaciones de interés en forma de posibilidades de interacción, control y configuración, ofreciendo un salto cualitativo en la visualización de la recursividad frente a las aplicaciones que existían hasta el momento.

Por otra parte, se ha evaluado la capacidad de SRec para ilustrar algoritmos de la técnica “divide y vencerás”. Se aprovechó la clasificación realizada en el apartado “2.8 Modelos Conceptuales de “Divide y Vencerás” en la Bibliografía” para estudiar a qué tipos de algoritmos se adaptan peor las vistas actuales de SRec. Se detectó que son soportados con limitaciones los algoritmos de carácter geométrico o los que tienen un carácter profundamente matemático. Ello permitió proponer una serie de ampliaciones y convenciones genéricas que ayudarían a SRec a la hora de ilustrar un mayor rango de algoritmos, lo que permitiría ampliar su capacidad, utilidad y tiempo de uso por parte de los profesores y alumnos, que accederían a una mejor representación de un mayor catálogo de problemas.

Capítulo 6. Evaluación de Usabilidad con Cuestionarios

Uno de los objetivos declarados al comienzo de este trabajo es la evaluación de la usabilidad de SRec. El primer apartado de este capítulo ofrece una visión general sobre qué es la usabilidad, mientras que el segundo da una perspectiva global sobre las pruebas de usabilidad. En el tercer apartado se detalla el método general empleado en todas las sesiones de evaluación de la usabilidad realizadas sobre SRec mientras que en el cuarto se recoge una evolución general de la aplicación y de sus valoraciones. A partir del quinto se describen y analizan las cinco sesiones de evaluación de usabilidad realizadas desde mayo de 2007 (momento de la primera prueba) hasta noviembre de 2010 (fecha en que tuvo lugar la quinta y última hasta el momento).

6.1 La Usabilidad

La usabilidad consta de múltiples definiciones, incluso la Organización Internacional para la Estandarización (conocida por sus siglas ISO) ofrece más de una definición dentro de sus normas de estandarización. Acogiéndonos a las utilizadas en las normas ISO/IEC 9126 (centrada en la evaluación del software) e ISO/IEC 9241 (que se centra en la ergonomía en el trabajo de personas con sistemas informáticos), podríamos decir que la usabilidad es la capacidad de ser comprendido, aprendido, usado así como de resultar atractivo para el usuario, traduciéndose en eficacia, eficiencia y satisfacción a la hora de alcanzar ciertos objetivos para una serie de usuarios concretos en determinadas condiciones de utilización.

La usabilidad se basa en seis pilares fundamentales [50], que dan pie a una serie de preguntas que se presentan justo a continuación de cada uno de ellos:

- Dificultad de aprendizaje: mide lo fácil que le resulta al usuario completar las tareas la primera vez que se coloca delante del sistema.
    - ¿Cuánta preparación previa necesitan recibir los usuarios?
    - ¿Qué documentación tienen disponible los usuarios?
    - Las preguntas que se hacen los usuarios, ¿están disponibles en la documentación?

- Eficiencia: una vez que han aprendido cómo utilizar la herramienta, ¿son eficientes los usuarios en la realización de la tarea respecto al tiempo?
    - ¿Qué quieren hacer los usuarios?
    - ¿Pueden hacer los usuarios las tareas a la velocidad esperada?
- Capacidad de ser recordado: estudia cuánto cuesta recordar a los usuarios cómo funciona el sistema tras estar un tiempo sin usarlo.
    - ¿Cometen los mismos errores o atraviesan las mismas dificultades que la primera vez que usaron la herramienta?
    - ¿Decrece significativamente la eficiencia tras pasar un tiempo sin utilizar la aplicación?
- Errores: es fundamental medir cuántos y qué errores cometen los usuarios, para conocer en qué medida "tropiezan" con la aplicación.
    - ¿Puede y sabe el usuario recuperarse de esos errores?
    - ¿Ayuda la aplicación a recuperarse de los errores?
    - ¿Generan perjuicios tales como pérdida de información esos errores?
- Satisfacción: intenta medir el placer resultante de emplear la herramienta para algún fin.
    - ¿Qué sensación le queda al usuario tras utilizar la aplicación?
    - ¿Volvería a utilizarlo sin estar obligado o condicionado?
- Utilidad: se evalúa si el sistema realmente hace lo que necesita el usuario.
    - ¿Se alcanzan todos los objetivos propuestos por el usuario?
    - Si no es así, ¿es porque no se sabe emplear la herramienta o porque esta no permite alcanzar tales metas?

Los beneficios de realizar un diseño contemplando la usabilidad son numerosos [51] y quedan detallados a continuación:

- Reducción del coste de aprendizaje: se consiguen sistemas fáciles de aprender, de carácter intuitivo y que hace uso de convenciones, por lo que se

alcanza antes el momento en el que se puede empezar a realizar trabajo productivo con autonomía.

- Disminución de la asistencia al usuario: el usuario es capaz de aprender y desenvolverse fácilmente con la aplicación, por lo que no necesita que ni la empresa propietaria o el profesor que le suministra el software le dediquen tiempo para que pueda emplear el programa.
- Disminución de los errores cometidos por el usuario: se traduce en un aumento de la eficiencia y de la motivación, con grandes beneficios para el estudiante, que ahorrará un tiempo valioso.
- Disminución de los costes de mantenimiento y rediseño: un sistema mal diseñado y que no es suficientemente usable, a la larga requiere tareas de modificación y mantenimiento mucho más pesadas, traumáticas y costosas que si el sistema fue diseñado desde el principio conservando los principios de la usabilidad.
- Aumento de la satisfacción y comodidad del usuario: en el contexto educativo es importante motivar a los alumnos, un sistema que les ayude facilita esa tarea pero un sistema que supone un obstáculo para alcanzar el objetivo logra el indeseable efecto contrario.
- Mejora la imagen y el prestigio del desarrollador: la realización de un software de calidad y usable ayuda a mejorar la imagen de la empresa o entidad que lo ha desarrollado, provocando el interés de terceros no sólo por ese sistema si no por otros desarrollados por los mismos autores o entidad.

Para poder cumplir los objetivos de usabilidad de una aplicación educativa se ha de ser consciente de que en la interacción entre las personas y los sistemas entran en juego múltiples disciplinas [51], como son:

- Documentación: la profesionalidad a la hora de generar documentación sobre la aplicación ayudará en gran medida a que los usuarios la encuentren útil cuando recurran a ella, encontrando y entendiendo la información que les suministra para poder aplicarla adecuadamente.

- Ergonomía: busca la comodidad del usuario y permite que éste logre pasar más tiempo trabajando con la aplicación y con menor esfuerzo, haciendo un uso realmente correcto del sistema.
- Informática: es fundamental conocer la tecnología disponible para emplearla y crear un sistema software sobre ella. La labor de diseño genera un modelo consistente que dará lugar a un sistema real aplicando metodologías profesionales de ingeniería que aseguren la robustez y calidad final del mismo.
- Psicología: explorar el comportamiento de la mente humana ayuda a diseñar mejor la interfaz y los procesos de tal forma que se interpreten y almacenen mejor por parte de la misma. Por ello, el modo de distribuir y representar la información debe ser adecuado para facilitar la asimilación por la mente humana.
- Sociología: estudia las características de comportamientos grupales (sociales, de carácter regional), intentando asociarlas a cuestiones ambientales, culturales y psicológicas y permite garantizar que un sistema está adaptado a lo que los usuarios de una determinada región esperan en cada momento.

En este trabajo durante todo el proceso se ha llevado a cabo un diseño centrado en el usuario:

- Conociendo al perfil de usuarios: se ha tenido en cuenta su perfil académico (por un lado, profesores, con amplios conocimientos de algoritmia, y por otro alumnos de segundo o tercer curso, según el plan de estudios, con conocimientos básicos previos de recursividad), la disciplina a la que pertenecen (la informática, por lo que se les presupone predispuestos a conocer nuevas herramientas informáticas, con experiencia ante el ordenador, conocedores de convenciones y estilos estándar…) y el contexto en el que van a usar la herramienta (breve periodo en un contexto docente, para una parte de una asignatura de algoritmia).
- Logrando que el sistema resuelva sus necesidades: los profesores necesitan una herramienta para ilustrar fácilmente un número ilimitado de ejemplos de

un catálogo indefinido de algoritmos y los alumnos una herramienta con la que poder analizar y depurar programas recursivos.

- Probando lo diseñado: se han realizado cinco evaluaciones de usabilidad de la herramienta entre alumnos a lo largo de todo el proceso iterativo realizado junto a periódicas evaluaciones de experto.

Durante el diseño de SRec se han tenido en cuenta diversos aspectos de la aplicación que afectan igualmente a la usabilidad de la misma, como son:

- Instalación sencilla: SRec se proporciona en forma de archivo único ejecutable que es capaz de auto-instalarse y quedarse enlazado mediante accesos directos.
- Facilidad de aprendizaje: se proporciona documentación (ayuda interactiva y manual) y se ha diseñado una interfaz intuitiva y sencilla.
- Eficiencia para la elaboración de trabajos (productividad): se ha enfocado el trabajo de diseño al objetivo de reducir al máximo la interacción necesaria para la generación de visualizaciones nuevas, reduciendo el esfuerzo del usuario.

Todo ello facilita y propicia que se pueda dar de una manera cómoda soporte a las actividades docentes exigibles durante la utilización de SRec.

6.2 Las Pruebas de Usabilidad

Las evaluaciones de usabilidad realizadas han constituido una parte amplia del trabajo que se presenta en esta tesis doctoral.

Se han realizado un total de cinco y éstas estuvieron enfocadas en medir el grado de satisfacción de los alumnos junto con la percepción de los mismos acerca de la calidad, facilidad de uso y capacidad de ilustración de la herramienta, aspectos relacionados respectivamente con los conceptos de satisfacción del usuario, eficiencia y eficacia dados en la definición de usabilidad en el apartado anterior.

Quedan recogidas en la Tabla 14 la fecha, la versión evaluada de SRec y la tarea, válida para la obtención de nota en la asignatura, que tuvieron que realizar los alumnos en las distintas evaluaciones. En las evaluaciones 4 y 5 aparecen respectivamente dos y tres tareas porque estas evaluaciones hicieron uso de dos y tres sesiones.

Eval.	Fecha	Tarea	SRec
1	24-5-2007	- Depurar un algoritmo erróneo	1.0
2	4-12-2007	- Eliminar redundancia en recursividad múltiple	1.0
3	14-11-2008	- Eliminar redundancia en recursividad múltiple	1.0
4	6-11-2009 4-12-2009	- Diseño de algoritmo recursivo lineal - Diseño de algoritmo “divide y vencerás”	1.2
5	7-10-2010 8-10-2010 15-10-2010	- Depurar un algoritmo erróneo - Depurar un algoritmo erróneo - Diseño de algoritmo “divide y vencerás”	1.2

Tabla 14. Evaluaciones realizadas.

6.3 Método General

Las cinco evaluaciones fueron realizadas en los laboratorios de los que dispone la Escuela Técnica Superior de Ingeniería Informática en el Campus de Móstoles de la Universidad Rey Juan Carlos. Cada alumno podía hacer uso de un ordenador, si bien las prácticas evaluables podían ser presentadas por parejas. Durante el transcurso de las sesiones los alumnos, que siempre pertenecieron a la carrera de Ingeniería Informática, podían realizar preguntas sobre el enunciado de la práctica y sobre el funcionamiento de la aplicación.

Las evaluaciones realizadas han tenido un esquema muy similar, si bien presentan algunas diferencias. Las evaluaciones 1, 2 y 3 estuvieron compuestas de una única sesión en el laboratorio, mientras que la evaluación 4 tomó dos sesiones y la 5 un total de tres. El esquema básico seguido en las sesiones de laboratorio, de dos horas de duración, es el siguiente:

- Demostración del profesor: se presenta la herramienta o una parte concreta de la misma para dejar ver cómo se interactúa y dar las líneas generales de actuación para el desarrollo de la práctica posterior. Esta parte suele tener una duración de entre 10 y 15 minutos.
- Actividades de familiarización: se proponen ejercicios que tienen como objetivo familiarizarse con la herramienta, con sus funcionalidades, con sus posibilidades de interacción y con sus opciones de configuración. Esta parte suele tener una duración de entre 35 y 40 minutos. Sólo tiene lugar si se trata de la primera sesión de una evaluación.
- Ejercicio didáctico, válido para la nota de la asignatura: se propone un ejercicio acorde con los conocimientos impartidos en clases teóricas previas,

que debe ser realizado con ayuda de SRec. Durante la realización de estos ejercicios en la quinta evaluación se realizaron observaciones sobre el comportamiento de los alumnos.

- Cuestionario: se les deja un cuestionario para que expresen una valoración numérica (valores enteros entre 1 y 5, ambos inclusive) sobre ciertos aspectos de SRec y ofrezcan su opinión (críticas, sugerencias…). En las evaluaciones 1 a 4 se realizó en formato papel, mientras que en la evaluación 5 tuvo un formato electrónico. Su duración se fija como máximo en 20 minutos.

A continuación se presenta en la Tabla 15 el transcurso de las cinco evaluaciones. Cada fila existente representa una sesión perteneciente a la evaluación correspondiente. La letra "D" significa Demostración del profesor; la letra "F", Actividades de Familiarización; la letra "E", Ejercicio evaluable; la letra "C", Cuestionario; el asterisco, versión breve; y la almohadilla, que se realizaron observaciones durante el transcurso de la actividad:

Evaluación 1	**Evaluación 2**	**Evaluación 3**	**Evaluación 4**	**Evaluación 5**
- D (15 min.) - F (35 min.) - E (40 min.) - C (20 min.)	- D (15 min.) - F (35 min.) - E (40 min.) - C (20 min.)	- D (10 min.) - F (40 min.) - E (45 min.) - C (15 min.)	- D (10 min.) - F (40 min.) - E (45 min.) - C* (15 min.)	- D (15 min.) - F (40 min.) - E# (55 min.)
			- D (10 min.) - E (80 min.) - C (20 min.)	- D (15 min.) - E# (95 min.)
				- E# (90 min.) - C (20 min.)

Tabla 15. Desarrollo esquemático de las sesiones de cada evaluación.

Se ha de hacer constar además que en las evaluaciones en que se realizó el cuestionario en formato papel no todos los alumnos lo entregaron (de manera intencionada o por descuido), por lo que difiere el número de participantes que se indican en algunas de las evaluaciones (contándose aquí el número de personas que realizaron la práctica de laboratorio) y el número de opiniones recogidas en los cuestionarios (resumido en el apartado "6.4 Evolución").

Por otra parte, en la quinta evaluación de usabilidad se recogieron otros productos tales como observaciones realizadas por expertos y archivos de registro de actividad de la aplicación para poder ampliar los estudios y obtener conclusiones más

detalladas así como una visión más amplia del estado de la aplicación con respecto a la usabilidad.

6.4 Evolución

Desde la primera evaluación (mayo de 2007) hasta la quinta y última (octubre de 2010) la evolución de SRec ha sido contundente, consiguiendo obtener una herramienta mucho más completa, útil, estable y adaptada a las necesidades de los usuarios.

De las 67 características identificables en SRec en la quinta evaluación, 21 de ellas (30,43%) estuvieron disponibles en la primera evaluación, 32 (47,76%) en la segunda, 42 (62,69%) en la tercera y 59 (88,05%) en la cuarta (el aumento respecto a la tercera evaluación fue elevado al haber un cambio de versión entre la 1.0 y la 1.2).

A continuación se presenta la Tabla 16 que recoge el listado de características a medida que fueron siendo incorporadas a la aplicación y examinadas en las evaluaciones de usabilidad llevadas a cabo. La columna de la derecha aporta el número de evaluación desde la que estuvo disponible tal característica.

Características	**Evaluación**
Soporte básico para recursividad con un único método	1
Capacidad de compilación y ejecución de código desarrollado por el usuario	1
Vista de árbol de activación	1
Vista de pila de control	1
Vista de traza	1
Vista de código	1
Controles básicos de animación	1
Capacidad de configuración de formato y tipografía	1
Configuración de coloreado de celdas por tipo de dato contenido	1
Configuración de coloreado y formateado del resto de elementos	1
Capacidad de configuración de cantidad de información mostrada	1
Configuración de datos mostrados en los nodos (entrada, salida, ambos)	1
Configuración de nodos históricos (atenuar, mantener, ocultar)	1
Configuración sobre mostrar o no subárboles saltados manualmente	1
Configuración manual de zoom	1
Gestión de archivos intermedios	1
Cargar y guardar visualizaciones	1
Generación de valores aleatorios para parámetros	1
Opción de cargar/guardar valores para parámetros	1
Editor de código simple	1
Sistema de ayuda navegable por hipervínculos	1
Capacidad de mostrarse en múltiples idiomas	2
Editar y reprocesar clase cargada	2
Información de errores de compilación al cargar clases	2
Información sobre excepciones durante la ejecución del algoritmo	2

Soporte básico para recursividad con más de un método involucrado	2
Configuración de visibilidad de métodos y parámetros antes de la creación de la visualización	2
Auto ajuste de zoom, incremento/decremento de zoom	2
Editor de código con acceso a mensajes del compilador	2
Información básica sobre nodos	2
Guardar animaciones GIF (árbol de activación)	2
Recuperación ante falta de memoria de Java al dibujar árboles grandes	2
Visor de navegación (vista global + detalle)	3
Mostrar/ocultar visor de navegación	3
Soporte para recursividad con más de un método involucrado, con capacidad de selección de información	3
Soporte para recursividad con métodos sin retorno	3
Gestión de la configuración	3
Almacenamiento de valores en cuadros de diálogo para parámetros durante la sesión	3
Guardar capturas únicas (formatos PNG/GIF/JPG)	3
Guardar capturas en serie (formatos PNG/GIF/JPG)	3
Guardar traza en formato HTML	3
Manual de usuario disponible (PDF)	3
Ligar la salida a la entrada en la vista cronológica	4
Mostrar estructura completa/parcial en vista cronológica	4
Configuración de visibilidad de métodos y parámetros antes de la creación y durante la visualización	4
Vista de estructura actual	4
Vista de árbol de activación orientada a estructura	4
Vista cronológica de estructura	4
Configuración de disposición de paneles	4
Aviso de trabajo no guardado al salir, si procede	4
Cargar animaciones GIF	4
Guardar animaciones GIF (cualquier vista)	4
Coloreado configurable de celdas por método de representación	4
Información general sobre la visualización en uso	4
Búsqueda y selección de llamadas	4
Soporte para la técnica Divide y Vencerás	4
Información detallada sobre nodos	4
Capacidad de activar nodo seleccionado	4
Archivo ejecutable auto instalador	4
Asistente gráfico para activación de vistas DYV	5
Iconos mejorados (>16 píxeles)	5
Editor de código integrado en la ventana principal	5
Editor de código con coloreado de sintaxis	5
Opción de creación de clases nuevas	5
Capacidad de registro de actividad	5
Búsqueda automática de Máquina Virtual de Java	5
Árboles colapsables durante el crecimiento	5

Tabla 16. Características y sesión de evaluación en la que se incorporaron.

Las características que se fueron añadiendo gradualmente estaban en gran parte recogidas o inspiradas en los cuestionarios de opinión que rellenaban los alumnos en las diferentes evaluaciones aportando críticas y sugerencias. A lo largo de las sesiones, se

apreció que las reivindicaciones y peticiones acerca de la herramienta se basaban en dos aspectos:

- La tarea: en función de la tarea que tuvieran que realizar (análisis, depuración, diseño o escritura de un algoritmo) y de los productos que se les exigiese (estudios, representaciones…), los alumnos realizaban unas peticiones u otras. Así, por ejemplo, cuando se les pedía representaciones gráficas y la aplicación no permitía la exportación, sólo entonces ésta fue ampliamente demandada; cuando se les pidió dibujar el grafo de dependencia de las llamadas, pidieron que SRec lo dibujase como una vista más; y cuando tuvieron que escribir un algoritmo desde cero, el editor de código fue la parte que más críticas y sugerencias recibió.
- Las funcionalidades ya implementadas: cuanto más desarrolladas están las funcionalidades del programa, más específicas son las peticiones que se realizan. Así, en la cuarta evaluación, el editor de código centró diversas críticas porque no estaba integrado en la ventana principal y no coloreaba la sintaxis mientras que en la quinta evaluación, con una tarea prácticamente idéntica pero con un editor integrado y con coloreado automático, las peticiones se tornaron más concretas, abarcando aspectos de tabulado, de edición inteligente (sugerencia de métodos, indicación de errores simultáneamente a la escritura…).

En cualquier caso, las críticas fueron tenidas en consideración sin que ello se tradujera en una mejora de los resultados. Como se podrá ver en los apartados siguientes, los resultados fueron bajando ligeramente a medida que se incorporaban funcionalidades, si bien hubo casos concretos en los que se logró invertir la tendencia negativa revisando los diseños y modos de funcionamiento.

Pregunta \ Evaluación	**1**	**2**	**3**	**4(1)**	**4(2)**	**5**	**M**
SRec es fácil de usar	3,88	4,50	4,20	4,39	3,90	3,94	4,15
Calidad general de SRec para analizar la recursividad (* técnica divide y vencerás)	3,38	4,29	4,00	-	-	3,84 *	3,96
SRec me ha gustado	3,63	4,26	3,95	4,07	3,85	3,84	3,97
Número de opiniones	7	28	21	28	19	49	

Tabla 17. Evolución de las puntuaciones de SRec sobre aspectos globales.

Se repasan en la Tabla 17 la progresión de las puntuaciones sobre aspectos generales, dando la media ponderada en la columna “M”. Se presenta también la Tabla

18 que recoge la evolución de las puntuaciones de diversos aspectos de SRec que fueron objeto de análisis por parte de los estudiantes. En las tres primeras evaluaciones se pidió una valoración general de las distintas partes de SRec (Columnas "G") mientras que en las evaluaciones 4 y 5 se pidieron valoraciones basándose en la calidad (columnas "C") por un lado y en la facilidad de uso (columnas "FU") por otro.

Aspecto \ Evaluación	**1 G**	**2 G**	**3 G**	**4(2) C**	**4(2) FU**	**5 C**	**5 FU**	**M**
Estructura del menú principal	3,75	4,07	-	4,11	3,95	3,78	4,00	3,94
Configuración de las visualizaciones	3,88	3,82	3,76	4,00	3,89	3,57	3,49	3,70
Interacción con los paneles (mover, scroll, …)	3,63	3,89	3,80	4,00	3,94	3,57	3,55	3,72
Controles de animación	4,00	4,50	-	4,22	3,94	3,96	3,96	4,08
Vista de traza	3,75	4,00	-	-	-	-	-	3,95
Vista de la pila de control	4,00	4,04	-	-	-	-	-	4,03
Vista de árbol de activación	4,25	4,43	4,00	4,11	4,22	3,82	4,00	4,06
Iconos	.	3,86	3,57	4,05	4,05	3,59	3,49	3,70
Configuración de zoom	-	-	3,71	3,71	3,71	3,24	3,33	3,44
Visor de árboles grandes	-	-	3,86	3,56	3,50	2,82	3,27	3,27
Generación de una visualización	-	-	4,00	4,05	4,16	4,22	4,20	4,16
Almacenamiento/cargado de visualización	-	-	4,14	4,26	4,21	3,73	3,71	3,90
Visualización almacenada en un fichero de captura	-	-	4,00	4,06	3,83	3,57	3,61	3,73
Vista cronológica	-	-	-	3,89	3,89	3,92	3,86	3,89
Vista de la estructura de datos	-	-	-	3,74	4,00	3,63	3,63	3,70
Control de la información que se muestra	-	-	-	3,61	3,61	3,35	3,51	3,48
Número de alumnos (172)	7	28	21	19		49		

Tabla 18. Evolución de las puntuaciones de SRec sobre aspectos concretos.

En general, las puntuaciones obtenidas por SRec en las evaluaciones 2 y 3 se sitúan por encima de la media global, cuando la aplicación había realizado mejoras sin añadir funcionalidades que aumentaran la complejidad del manejo, mientras que las de la evaluación 4 y, sobre todo la 5, se sitúan por debajo de la media, cuando SRec incorporó nuevas funcionalidades para la interacción, presentación de información, visualización de la técnica "divide y vencerás" y edición de código Java.

A pesar de todo, la valoración que se realiza es positiva, pues SRec es hoy una aplicación útil, versátil, que oferta diversas funcionalidades que son utilizadas tanto por los profesores como por los estudiantes y que cumple su objetivo de ayudar en el análisis y la depuración de programas recursivos. Se mantendrá, no obstante, como trabajo futuro realizar mejoras en la interfaz con el fin de conseguir una perfecta armonía entre funcionalidad y usabilidad.

6.5 Evaluación de Usabilidad nº 1

La primera de las evaluaciones de usabilidad de SRec tuvo lugar el 24 de mayo de 2007. Se contó para ella con la participación de 7 alumnos de la asignatura optativa "Estructura de datos y algoritmos avanzados" de 3º curso de Ingeniería Informática, impartida por el profesor Carlos A. Lázaro Carrascosa.

Se les planteó como ejercicio final de la práctica la depuración de una versión del algoritmo de ordenación por mezcla, que contenía dos errores.

Los resultados de la evaluación provienen desde el cuestionario que los 7 alumnos rellenaron al finalizar la sesión de trabajo con la aplicación. Este cuestionario les planteó preguntas de respuesta numérica sobre aspectos generales y concretos de SRec, dejándoles responder también de manera abierta para aportar críticas y sugerencias de mejora.

6.5.1 Valoraciones Numéricas

En primer lugar se presenta la Tabla 19, que contiene los valores medios de las respuestas numéricas dadas a cuestiones generales.

Cuestión	**Media**	**Desv. T.**
SRec me ha ayudado a analizar algoritmos para comprobar que la solución propuesta es correcta	4,13	0,45
SRec es fácil de usar	3,88	0,49
SRec me ha gustado	3,63	0,35
Calidad general de SRec para analizar la recursividad	3,38	0,76
SRec me ha ayudado a analizar algoritmos y a encontrar el error	2,63	1,07

Tabla 19. Puntuaciones de aspectos globales en la primera evaluación.

Se puede observar que la mejor puntuación la obtiene la cuestión que pregunta acerca de si SRec ha permitido analizar la solución propuesta para ver si era correcta. La mayor variabilidad de respuestas se da en la pregunta que obtiene una nota media más baja, la que pregunta acerca de si SRec ha conseguido ofrecer ayuda para encontrar el error existente en el código dado. Se recogen en la Tabla 20 los valores medios de las respuestas numéricas sobre aspectos concretos de SRec.

Las mejores puntuaciones son obtenidas por los controles de animación, la vista de la pila de control, y en primer lugar, el árbol de activación, mientras que el modelo de interacción con los paneles obtiene la puntuación más baja de todas.

Cuestión	Media	Desv. T.
Vista de árbol de activación	4,25	0,35
Controles de animación	4,00	0
Vista de la pila de control	4,00	0,49
Configuración de las visualizaciones	3,88	0,49
Estructura del menú principal	3,75	0,45
Vista de traza	3,75	0,70
Interacción con los paneles (mover, scroll, zoom…)	3,63	0,64
Media de las puntuaciones anteriores	3,89	

Tabla 20. Puntuaciones de aspectos concretos en la primera evaluación.

La Ilustración 46 muestra el gráfico que alberga las medias individuales de las puntuaciones aportadas por cada alumno para SRec:

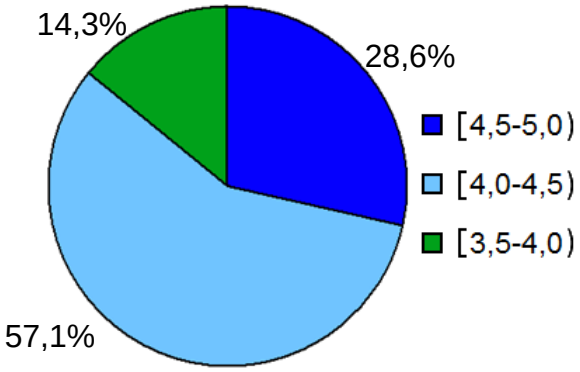


Ilustración 46. Aceptación personal de SRec (1ª evaluación)

Tal y como se puede ver en el gráfico, la primera versión evaluada de SRec obtuvo una buena aceptación. Así, más del 28% de los alumnos puntuaron a SRec con una nota media superior al 4,5 sobre 5, mientras que el 57% le situó entre el 4 y el 4,5.

6.5.2 Críticas y Sugerencias Recibidas

La característica que más gustó a los alumnos fue la facilidad del manejo de la herramienta. El hecho de que con apenas un par de clics se puedan generar visualizaciones útiles y con calidad gráfica fue uno de los factores que más ayudó a lograr la aceptación de la herramienta entre los alumnos. La vista del árbol de recursión, que les permitía ver de una manera clara la ejecución completa el algoritmo que estaban visualizando, fue el segundo elemento que más impacto positivo tuvo en los alumnos.

Por detrás quedan otros factores como la claridad y utilidad de las visualizaciones en general y la capacidad de personalización de las vistas que ofrece la aplicación. Algunos alumnos también se animaron a valorar positivamente las vistas de

la traza y de la pila de control, que complementaban la vista principal. También hubo quien destacó que se podía ver fácilmente el resultado de cada subllamada y de la ejecución total del algoritmo, lo cual le resultaba de interés.

Respecto a las sugerencias sobre SRec, las más destacadas son las que echan en falta el seguimiento de la ejecución a través del panel de código, remarcando las sentencias que se van ejecutando. Sin embargo, la aplicación no funciona por sentencias, sino por subllamadas, por lo que el enfoque del resaltado de sentencias no se adapta al funcionamiento del programa ya que supondría la modificación del paso de animación. La inclusión de valores de variables en la vista de traza sigue un planteamiento similar y requeriría igualmente que la aplicación avanzase por sentencias, en lugar de por subllamadas recursivas como es su funcionamiento actual.

Otros alumnos señalaron la dificultad que presentaba ver árboles de gran tamaño, ya que se perdían por él. También se destacó por parte de los alumnos que el programa no ayudaba activamente a encontrar los errores que contenían los códigos suministrados en el enunciado de la práctica. Esto es fruto de que la aplicación no ha sido concebida como un depurador sino como una herramienta para el análisis de algoritmos correctos.

Por último, algunos alumnos indicaron que el funcionamiento del programa podría ser más cómodo para el usuario si incluyese un botón de recarga de una clase para cuando ésta ha sido modificada por el usuario. De esta forma, se evita tener que buscarla por el sistema de ficheros cada vez que se quiere volver a cargar. El botón fue añadido posteriormente para agilizar la carga de clases.

A la pregunta que pedía la mención de características que SRec no tenía pero que sería interesante que incorporara, casi la mitad de las respuestas solicitaron el resalte dinámico de instrucciones de código a medida que se iba avanzando paso a paso en la visualización. Otras peticiones fueron el resalte de ciertos nodos (implementado posteriormente en la versión 1.1), más opciones en la vista de traza, resalte de errores sintácticos en el sencillo editor que proporciona SRec, la modificación de las combinaciones de teclado para el acceso rápido a las funcionalidades de SRec o más opciones en el desarrollo de visualizaciones (si bien no se especificó ninguna funcionalidad concreta).

6.5.3 Conclusiones

Los resultados obtenidos en esta primera evaluación fueron apreciablemente mejorables, sin embargo, no pueden ser considerados en ningún caso malos, pues la evaluación se realizó tras apenas 8 meses de trabajo a tiempo parcial, con un número muy limitado de funcionalidades y con algún que otro error de ejecución que fue corregido posteriormente, lo que constituyó más una versión preliminar que una versión plenamente estable y usable.

En general, los resultados pueden ser entendidos como la aceptación de las visualizaciones de algoritmos en el aula para la ayuda en tareas como el análisis.

Uno de los aspectos más útiles de la evaluación fue, sin lugar a dudas, la recogida de sugerencias y críticas que marcaron fuertemente el diseño e implementación posteriores. El alto número de sugerencias era de esperar al encontrarse la aplicación en un estado aún de corto desarrollo.

6.6 Evaluación de Usabilidad nº 2

La segunda de las evaluaciones de usabilidad de SRec tuvo lugar el 4 de diciembre de 2007. Se contó para ella con la participación de 28 alumnos de la asignatura “Diseño y análisis de algoritmos”, asignatura obligatoria de 3º de Ingeniería Informática impartida por el Doctor J. Ángel Velázquez Iturbide.

Como ejercicio final se le pidió a los alumnos que analizaran el algoritmo de recursividad múltiple del problema de competición para entregar posteriormente dos representaciones gráficas: una del árbol de recursión (copiada desde la pantalla de SRec) y otra del grafo de dependencia.

Los resultados de la evaluación quedaron recogidos gracias al cuestionario que los 28 alumnos completaron en los minutos finales de la sesión en el laboratorio.

6.6.1 Valoraciones Numéricas

La Tabla 21 muestra las diversas calificaciones que obtuvo SRec en aquellas preguntas que preguntaban sobre capacidades generales u opiniones globales. Se puede observar que la mejor puntuación la obtiene la cuestión que pregunta acerca de si SRec es fácil de usar con una nota considerablemente alta. Ésta es la demostración de que en

este momento el camino que se estaba recorriendo era el adecuado, pues la usabilidad se identificaba como una de las características de la aplicación.

Cuestión	Media	Desv. T.
SRec es fácil de usar	4,50	0,50
SRec me ha ayudado a identificar la dependencia de llamadas recursivas	4,36	0,55
Calidad general de SRec para analizar la recursividad	4,29	0,45
SRec me ha ayudado a analizar algoritmos para estudiar qué llamadas se realizan en tiempo de ejecución	4,29	0,59
SRec me ha gustado	4,26	0,44

Tabla 21. Puntuaciones de aspectos globales en la segunda evaluación.

La mayor variabilidad de respuestas se da en las dos preguntas sobre la ayuda prestada para estudiar qué llamadas se realizan en tiempo de ejecución. Se presenta ahora la Tabla 22 que contiene los valores medios y la desviación típica de las respuestas numéricas acerca de partes concretas de SRec.

Cuestión	Media	Desv. T.
Controles de animación	4,50	0,57
Vista de árbol de activación	4,43	0,73
Estructura del menú principal	4,07	0,70
Vista de la pila de control	4,04	0,73
Vista de traza	4,00	0,76
Interacción con los paneles (mover, scroll, zoom…)	3,89	0,77
Iconos	3,86	0,87
Configuración de las visualizaciones	3,82	0,80
Media de las puntuaciones anteriores	4,07	

Tabla 22. Puntuaciones de aspectos concretos en la segunda evaluación.

Las mejores puntuaciones son obtenidas por los controles de animación y la vista de la pila del árbol de activación, mientras que el modelo de configuración de las vista obtiene la puntuación más baja de todas, que a su vez es de las que presenta una mayor variabilidad en las respuestas junto a la que pregunta acerca de los iconos, que también se encuentra entre las puntuaciones más bajas.

Atendiendo a las medias individuales de las puntuaciones aportadas por cada alumno, se muestra en la Ilustración 47 la valoración obtenida de SRec.

El porcentaje de estudiantes que puntuaron a SRec con una nota mayor a 4,5 alcanza el 21% (7 puntos menos que en la anterior sesión), mientras que el de alumnos que puntuaron a SRec con entre 4 y 4,5 puntos supera el 37%, haciéndose significativo (y mayoritario) el aumento del grupo de alumnos que puntúa a SRec con una media situada en entre 3,5 y 4 puntos, que asciende al 39%.

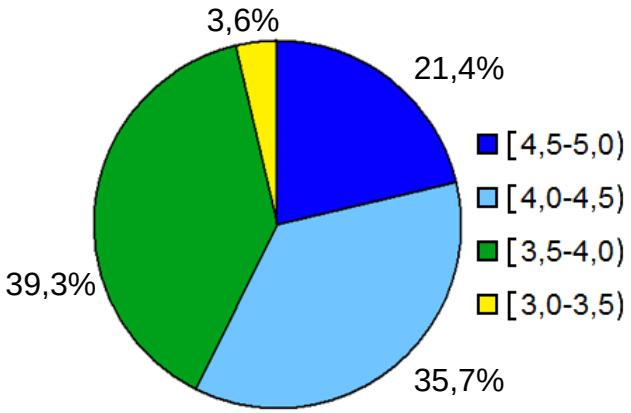


Ilustración 47. Aceptación personal de SRec (2ª evaluación)

6.6.2 Críticas y Sugerencias Recibidas

En esta ocasión la herramienta también consiguió un alto éxito de aceptación entre los alumnos, si bien el mayor número de participantes en la sesión proporcionó una mayor variedad de opiniones y sugerencias que en la anterior sesión.

La cualidad más resaltada por los estudiantes fue la claridad y utilidad de las vistas proporcionadas por SRec. Así, explicaron abiertamente que las visualizaciones realmente les ayudaban en el proceso de comprensión y análisis del funcionamiento de los algoritmos, objetivo fundamental del programa. La vista del árbol de activación fue también muy bien recibida por los alumnos, siendo el segundo elemento del programa mejor aceptado. Dos tercios de los individuos encuestados respaldaron una de estas dos opiniones expuestas.

A continuación, la vista de la pila de control se sitúa como el siguiente elemento con mayor aceptación, vista que complementa la información mostrada en el árbol de activación. Con un nivel de aprobación similar se encuentran las opciones de manejo del programa, calificadas como sencillas por gran parte del alumnado. Ya en menor medida, también contaron con cierto respaldo la vista de la traza y la capacidad de aprendizaje que proporciona SRec gracias a la visión conjunta de las vistas proporcionadas, todas ellas complementarias y siempre con un funcionamiento sincronizado.

También recibieron mención la flexibilidad en la configuración de la aplicación y la funcionalidad de edición de código. En el lado negativo, las funcionalidades que más críticas recibieron fueron la exportación de una visualización a GIF animado y la vista de traza. El proceso de exportación a GIF animado exigía en esta versión mantener

la ventana visible mientras se realizaba el guardado de la animación, un proceso que, en visualizaciones grandes, podía tomar algún que otro minuto, de tal forma que no se podía emplear el ordenador para otras tareas de manera concurrente. Este proceso fue optimizado en siguientes versiones, eliminando esa restricción. Por otro lado, varios alumnos expresaron un ligero rechazo a la vista de traza, fundamentalmente por considerarla redundante.

De nuevo varios alumnos volvieron a mencionar la dificultad que podía llegar a tener la visualización de árboles muy grandes, ya que por entonces aún se contaba con la versión 1.0 del programa, que no contenía ningún visor de navegación. También quedó mencionado que la aplicación podía tardar algunos segundos en abrir una visualización con un árbol de activación muy grande. Otras cuestiones objeto de crítica fueron el alto número de opciones de configuración disponibles, la poca flexibilidad de la barra de herramientas, la falta de libre movilidad de los paneles que contenían las vistas y el hecho de que el panel de código redujera el espacio disponible que se podía destinar a las vistas.

Como principales sugerencias recibidas, destaca de nuevo la petición de incluir en la vista del código de la clase el resaltado de las instrucciones que se van ejecutando. También destacaron las peticiones de que se admitieran otros lenguajes de entrada para la codificación de los algoritmos como C. Además, fruto de los requisitos del enunciado, nació una nueva demanda, la de la representación del grafo de dependencias, para que complementara las vistas existentes.

Por entonces, la aplicación sólo podía exportar animaciones GIF, de ahí que ante la petición en el enunciado de que aportaran en su informe de prácticas un árbol de activación, pidieran que la aplicación pudiera realizar capturas estáticas de la vista del árbol de activación en formato JPG.

Otras sugerencias recibidas fueron la modificación de algunos atajos de teclado, la funcionalidad de auto zoom según va avanzando la animación de la visualización, mayores opciones de movilidad de los paneles, la creación de un visor de navegación, información sobre si la ejecución del algoritmo levantó alguna excepción (se implementó posteriormente), cálculo de la complejidad del algoritmo en tiempo y memoria (actualmente la aplicación ofrece el número total de subllamadas realizadas), funcionalidad de imprimir en papel el contenido de las vistas, aportación como nueva

vista de un diagrama de flujo, “inserción en el árbol de la operación que realiza el algoritmo” y la posibilidad de “auto formato en la vista del árbol de activación”.

6.6.3 Conclusiones

Los datos suponen una mejora notable en las valoraciones numéricas dadas por los alumnos, pues la mayoría de ellas queda situada por encima del 4 dentro de una escala del 1 al 5. Hay dos motivos que pueden explicarlos:

- Diseño centrado en el usuario: se tuvieron muy en cuenta las sugerencias recibidas en la primera evaluación, por lo que algunas de las quejas ya no podían seguir teniendo lugar.
- La versión era más estable: a diferencia de la primera versión, de carácter preliminar, en esta evaluación se contó con una versión más depurada y menos propensa a sufrir errores durante su funcionamiento.

De esta manera, se logran unos resultados ampliamente satisfactorios que animan a seguir el camino emprendido, manteniendo la usabilidad como una de las constantes del trabajo que debe realizarse. Además, se comenzaron a recibir los primeros comentarios espontáneos en los informes de prácticas acerca de la percepción de que SRec les había ayudado a conseguir realizar la tarea propuesta.

6.7 Evaluación de Usabilidad nº 3

La tercera sesión de evaluación de usabilidad se desarrolló el 14 de noviembre de 2008. De nuevo los participantes, 33 en total, eran alumnos de 3º curso de la asignatura obligatoria “Diseño y análisis de algoritmos”, impartida por el profesor J. Ángel Velázquez Iturbide. Se mantuvo la estructura de la sesión, por lo que fue muy parecida a la de las pruebas anteriores.

Tras los ejercicios de familiarización con la herramienta, en esta tercera sesión se les planteó un problema donde tenían que eliminar la redundancia debida a la recursividad múltiple empleando técnicas de memorización y tabulación, ofreciendo posteriormente además un cálculo de la complejidad de los algoritmos resultantes. Este ejercicio requería las mismas tareas que el de la segunda evaluación sobre un algoritmo de complejidad muy similar.

Adicionalmente tenían que dibujar un árbol de recursión (la aplicación ya les permitía sacar una imagen capturada del mismo) y su grafo de dependencia asociado. Los datos que recogen los siguientes apartados están extraídos desde los cuestionarios que entregaron 22 alumnos.

6.7.1 Valoraciones Numéricas

Una vez más, se pueden desglosar en dos tablas las valoraciones numéricas conseguidas por SRec. La primera de ellas, la Tabla 23, contiene las puntuaciones que obtuvo SRec en las preguntas sobre capacidades generales u opiniones globales.

Cuestión	**Media**	**Desv. T.**
SRec es fácil de usar	4,20	0,41
SRec me ha ayudado a analizar algoritmos para estudiar qué llamadas se realizan en tiempo de ejecución	4,19	0,50
SRec me ha ayudado a identificar la dependencia de llamadas recursivas	4,04	0,78
Calidad general de SRec para analizar la recursividad	4,00	0,69
SRec me ha gustado	3,95	0,49

Tabla 23. Puntuaciones de aspectos globales en la tercera evaluación.

Se puede observar que la mejor puntuación, que aun siendo inferior a la evaluación número 2 mantiene un buen valor por encima del 4, la sigue consiguiendo la cuestión que pregunta acerca de si SRec es fácil de usar, lo que confirma que la usabilidad se identificaba como una de las características de la aplicación. La mayor variabilidad de respuestas se da en la pregunta sobre la ayuda prestada para identificar la dependencia de llamadas recursivas.

Se presentan en la Tabla 24 los valores medios de las respuestas numéricas acerca de partes concretas de SRec.

Cuestión	**Media**	**Desv. T.**
Proceso de almacenar/cargar una animación	4,14	0,64
Vista de árbol de activación	4,00	0,61
Visualización almacenada en un fichero de captura	4,00	0,62
Proceso de generación de una animación	4,00	0,76
Visor de árboles grandes	3,86	0,83
Interacción con los paneles (mover, scroll, zoom…)	3,80	0,81
Configuración de las visualizaciones	3,76	0,75
Controles de animación	3,71	0,70
Configuración de zoom	3,71	1,03
Iconos	3,57	0,85
Media de las puntuaciones anteriores	3,85	

Tabla 24. Puntuaciones de aspectos concretos en la tercera evaluación.

La más alta puntuación la consigue el proceso de carga y guardado de visualizaciones, una característica muy importante para profesores y alumnos, que permite trabajar en múltiples sesiones con el mismo algoritmo y valores de entrada sin ningún esfuerzo. Por su parte, la configuración de zoom es la que mayor variabilidad de valores consiguió.

Atendiendo a las medias individuales de las puntuaciones aportadas por cada alumno, se muestra en la Ilustración 48 la valoración obtenida de SRec:

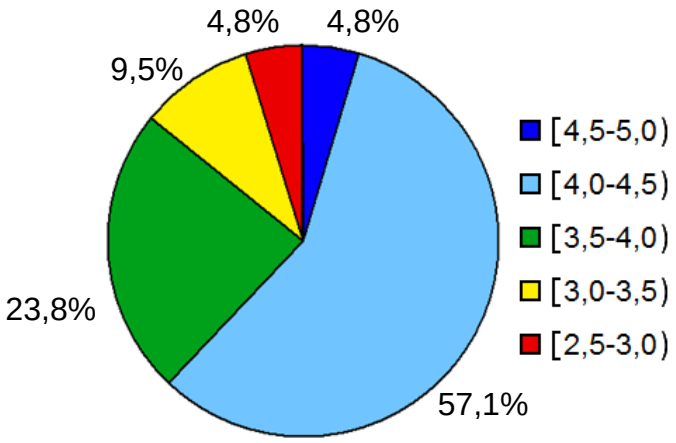


Ilustración 48. Aceptación personal de SRec (3ª evaluación)

A pesar de que se mantiene la tendencia decreciente del grupo que puntúa a SRec con más de 4,5 puntos, que se reduce a casi un 5%, la proporción total de personas que puntuaron a SRec con una nota superior a 4 ascendió hasta el 62% (en la anterior sesión se mantuvo en el 58%), gracias a que más del 57% de los alumnos colocó a SRec una nota media de entre el 4 y el 4,5.

Casi el 24% de los encuestados dio como nota media de SRec un valor entre 3,5 y 4 puntos, mientras que sólo el 14% restante dio una nota por debajo del 3,5, y siempre superior al 2,5 (nivel de aprobado).

6.7.2 Críticas y Sugerencias Recibidas

En esta ocasión la cantidad de respuestas abiertas contestadas fue menor, tanto en aquellas que pedían valorar los aspectos positivos como en las de los aspectos negativos. Aun así, se pudo recoger información de interés sobre los puntos fuertes de la aplicación o las carencias que los alumnos detectaron en SRec.

Al igual que en la segunda sesión, la principal cualidad de SRec destacada por los alumnos fue la alta claridad y utilidad de las vistas ofrecidas, haciendo de la aplicación una herramienta de calidad para su periodo de aprendizaje. La facilidad de manejo, una cualidad que se intentó mantener a toda costa pese a la cada vez mayor cantidad de funcionalidades, y la vista del árbol de recursión fueron nuevamente dos de las cualidades mejor valoradas por los alumnos.

Otros elementos de SRec que fueron mencionados positivamente son: la vista de traza, la herramienta de zoom, la capacidad de exportación a formato GIF animado, el proceso de carga de clases, y la posibilidad de guardar las visualizaciones en formato XML para recuperarlas en sesiones posteriores.

En el polo opuesto, la ingente cantidad de opciones de configuración pareció no agradar a algunos alumnos, que destacaron esto como un aspecto negativo por primera vez, bien por considerarlo irrelevante o bien por resultar demasiado complejo. El proceso de exportación a GIF animado (por el hecho de no ser instantáneo y no permitir su cancelación) recibió alguna crítica, igual que la visión de árboles grandes, que pese a contar ya con un visor de navegación, posibilidades de zoom y de desplazamiento por la vista, algunos alumnos pudieron experimentar alguna dificultad.

Otras críticas se enfocaron en el tiempo de ejecución y creación de visualizaciones en el caso de árboles grandes, la barra de herramientas por su poca flexibilidad de ubicación en la ventana, el proceso de carga de clases y la interfaz general, a la que se proponía realizar cambios pero sin especificar ninguno.

Como principal reivindicación apareció la generación de un grafo de dependencias, comprensible al ser una representación requerida en el ejercicio evaluable que realizaron durante la sesión y que no les aportaba el programa. La segunda petición fue la de un cambio en los iconos que representan las diferentes funcionalidades de SRec tanto en los menús como en la barra de herramientas, pues al parecer creaban confusión y resultaban poco intuitivos.

Otras peticiones, menos secundadas que las anteriores, fueron la de que SRec se pudiera ejecutar en otras plataformas distintas a Windows, la adición de nuevas opciones de visualización, como la de pantalla completa o la exportación de capturas a gran tamaño, posibilidad que añadió en versiones posteriores.

6.7.3 Conclusiones

Con esta evaluación se pudo comprobar que un aumento de las funcionalidades no tiene porqué derivar ni en una mayor aceptación ni tampoco en una mayor facilidad de uso.

De hecho, está ampliamente aceptado que cuando una aplicación alberga un elevado número de funcionalidades comience a ser percibida como una herramienta grande, cuyo trabajo de aprendizaje debe ser elevado para poder manejarla adecuadamente en todas las situaciones que puedan darse, por lo que la simplicidad de la interfaz es una característica fundamental [51] para minimizar el impacto del aumento de la funcionalidad.

Esta evaluación sirvió, por tanto, para comenzar a percibir que, aun incorporando funcionalidades adecuadas y necesarias, es necesario que la interfaz no se resienta volviéndose más compleja o menos intuitiva para no aumentar la curva de aprendizaje, aspecto fundamental en una aplicación tan específica que se emplea en un contexto en el que el tiempo es un recurso escaso valiosísimo y durante un periodo corto de utilización (durante varios temas de una única asignatura). Una de las medidas que se tomaron fue reorganizar la estructura de los menús para hacer más fácil encontrar la funcionalidad buscada.

6.8 Evaluación de Usabilidad nº 4

Durante el curso académico 2009/2010 tuvo lugar la cuarta evaluación sobre SRec 1.2, una versión que incluye importantes mejoras frente a la versión 1.0, como las opciones de interacción introducidas en las vistas desde la versión 1.1 o la integración de una segunda técnica de diseño soportada por la aplicación, “divide y vencerás”.

A pesar del largo camino ya andado en la implementación del sistema, seguía resultando muy conveniente conocer si ese camino se mantenía en la dirección adecuada o si por el contrario era incorrecto el rumbo que llevaba SRec. Esta sesión, por tanto, nos permitió saber si aspectos nuevos tuvieron un desarrollo adecuado para el posterior uso de la herramienta.

Esta evaluación se celebró mediante dos sesiones fechadas en días diferentes. El primer día de la evaluación fue el 6 de noviembre de 2009, mientras que el segundo

tuvo lugar el 4 de diciembre de 2009. Ambas se celebraron en el mismo laboratorio y horario semanal. Los experimentos de ambos días siguieron el mismo esquema de actuación que en evaluaciones anteriores por parte del profesorado y de los alumnos.

Se contó para esta cuarta evaluación en su primer día con la participación de 28 alumnos de la asignatura obligatoria “Diseño y análisis de algoritmos”, impartida por el profesor J. Ángel Velázquez Iturbide, mientras que el segundo día participaron 27.

Se les planteó como práctica el primer día la implementación de un método recursivo que resolviera el problema de los anillos de los números, entendiendo “anillo” como el número de trazos cerrados que tiene la representación escrita de un número, el número de entrada debía poder constar de varias cifras. El segundo día los alumnos hicieron uso de las capacidades de SRec para visualizar algoritmos diseñados bajo la técnica “divide y vencerás”, lo que requirió un manejo más avanzado de la herramienta. El ejercicio consistió en construir un algoritmo que calculase los dígitos comunes de un vector de números.

En las dos sesiones se les proporcionó un cuestionario que todos los alumnos rellenaron y entregaron al final de las mismas. Los datos de estos cuestionarios aparecen expuestos a continuación.

6.8.1 Valoraciones Numéricas

En cada una de las dos sesiones de laboratorio se pasó un cuestionario a los alumnos: el de la primera sesión tan sólo recogía una percepción general de la aplicación mientras que el segundo era mucho más detallado, recorriendo diversos aspectos de SRec y pidiendo, por primera vez, dos valoraciones para ellos, una referida a la calidad y otra a la facilidad de manejo.

Cuestión	**Media**	**Desv. T.**
SRec es fácil de usar	4,39	0,68
SRec me ha ayudado a ilustrar la definición inductiva del algoritmo	4,32	0,63
SRec incluye las funciones adecuadas para ilustrar el comportamiento de los algoritmos recursivos	4,25	0,67
SRec me ha gustado	4,07	0,78

Tabla 25. Puntuaciones de aspectos globales de la cuarta evaluación (día 1).

Se presenta en primer lugar la Tabla 25, que contiene los valores recogidos el primer día, cuando los alumnos sólo hicieron uso de las características relativas a la visualización de la recursividad general. Se puede apreciar cierto repunte en las

valoraciones respecto a la tercera sesión, obteniendo en todos los casos una puntuación por encima de 4, destacando, una vez más, la pregunta acerca de la facilidad de uso, que quedó muy cerca de su mejor registro (4,50 en la segunda sesión).

Atendiendo a las medias individuales de las puntuaciones aportadas por cada alumno, se muestra en la Ilustración 49 la valoración obtenida de SRec sobre 5:

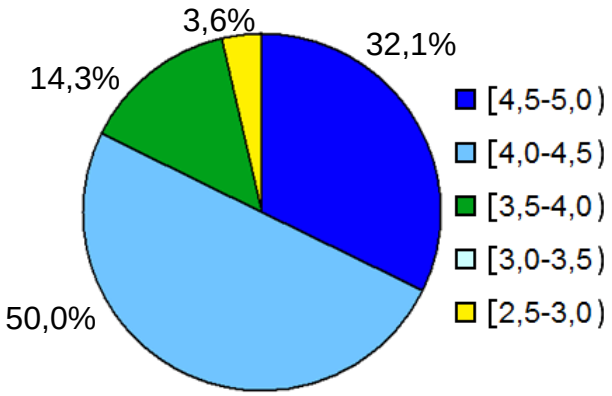


Ilustración 49. Aceptación personal de SRec (4ª evaluación, día 1)

Tal y como se puede ver en el gráfico, la aceptación de SRec fue altamente elevada. Así, un 32% de los asistentes (casi un tercio) le otorgó un nota media entre el 4,5 y el 5, mientras que un 50% lo situó entre el 4 y el 4.5, lo que supone que el 82% de los estudiantes aprobaron a SRec con un notable alto o sobresaliente (notas entre el 4 y el 5). Atendiendo a este dato, esta es la segunda mejor valoración de SRec de entre todas las que se han realizado hasta el momento acerca de la satisfacción personal. Un 14% de los estudiantes le puso una nota media entre 3.5 y 4, haciendo residual el grupo 2.5-3.0, con sólo un representante (3,5%).

Seguidamente se presentan los datos numéricos de la segunda sesión de laboratorio de esta cuarta evaluación. Los principales cambios vinieron porque los alumnos tuvieron que trabajar por primera vez con las nuevas vistas orientadas a ilustrar la técnica de diseño “divide y vencerás”: el proceso de cargado de clases requería ahora de la interacción del usuario y ahora pasaban a manejar seis vistas en lugar de tres. Todo ello hacía que el manejo del programa se tornara algo más complejo, lo que se dejó notar sensiblemente en los resultados. Éstos aparecen en la Tabla 26 para las cuestiones generales y en la Tabla 27 para las partes específicas de la aplicación.

Estos datos son sensiblemente inferiores a los recogidos en la primera sesión. La pregunta “SRec es fácil de usar” pierde 0,49 puntos, la pregunta “SRec me ha ayudado a ilustrar la definición inductiva del algoritmo” baja igualmente 0,49 puntos, y la pregunta

"SRec me ha gustado" pierde 0,22 puntos. Sobre la capacidad y calidad para ilustrar algoritmos, la puntuación para los algoritmos recursivos es de 4,25, mientras que para los de "divide y vencerás" pierde 0,31 puntos, hasta quedarse en 3,94.

Cuestión	Media	Desv. T.
Calidad general de SRec para ilustrar algoritmos de Divide y Vencerás	3,94	0,78
SRec es fácil de usar	3,90	0,83
SRec me ha gustado	3,85	0,77
SRec me ha ayudado a ilustrar la definición inductiva del algoritmo	3,83	0,76
SRec me ha ayudado a ilustrar el comportamiento detallado del algoritmo	3,68	1,08

Tabla 26. Puntuaciones de aspectos globales de la cuarta evaluación (día 2).

A continuación se exponen en la Tabla 27 los datos medios cosechados en la segunda sesión acerca de parte concretas de la aplicación, tanto sobre calidad (columna "C") como sobre facilidad de uso (columna "FU"), así como sus respectivas desviaciones típicas ("DT").

Cuestión	C	DTC	FU	DTFU
Estructura del menú principal	4,11	0,71	3,95	0,69
Iconos	4,05	1,10	4,05	0,89
Vista de árbol de activación	4,11	0,74	4,22	0,71
Visor de árboles grandes	3,56	1,07	3,50	1,17
Vista cronológica de la estructura	3,89	0,87	3,89	0,94
Vista de la estructura de datos	3,74	0,64	4,00	0,73
Control de zoom	3,71	1,02	3,71	1,02
Calidad del control de información que se muestra	3,61	1,06	3,61	0,95
Configuración de formatos gráficos	4,00	0,82	3,89	0,74
Controles de animación	4,22	0,71	3,94	0,85
Interacción con los paneles (mover, scroll, zoom…)	4,00	0,89	3,94	0,97
Proceso de generación de una animación	4,05	0,89	4,16	0,93
Proceso de almacenar/cargar una animación	4,26	0,85	4,21	0,95
Visualización almacenada en un fichero de captura	4,06	0,78	3,83	1,01
Media de las puntuaciones anteriores	3,95		3,92	

Tabla 27. Puntuaciones de aspectos concretos de la cuarta evaluación (día 2).

Los controles de animación y los procesos de carga y guardado de visualizaciones en formato XML son las partes que más calidad tienen según los estudiantes, mientras que el visor de árboles grandes y la configuración de la cantidad de información que se muestra obtienen los peores resultados en este aspecto.

Por su parte, en los que respecta a facilidad de uso, las mejores puntuaciones las obtienen la vista del árbol de activación y el ya citado proceso de carga y guardado de visualizaciones en formato XML. En el polo opuesto se encuentran nuevamente que el visor de árboles grandes y la configuración de la cantidad de información que se muestra.

Se repasa mediante la Ilustración 50 la nota media obtenida por el programa (siempre sobre 5) otorgada por cada uno de los participantes, donde se observa que aparece un nuevo grupo, que hasta ahora nunca se había tenido que contemplar en esta medición, el del rango [2.0-2.5).

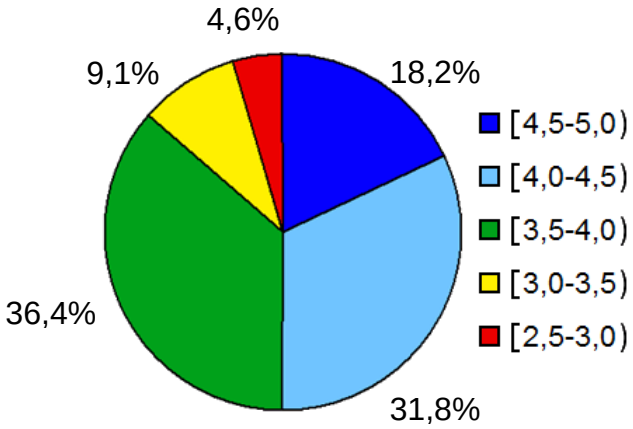


Ilustración 50. Aceptación personal de SRec (4ª evaluación, día 2)

El porcentaje de estudiantes que puntuaron a SRec con una nota mayor o igual a 4,5 alcanza el 18,2% (frente al 32% registrado el primer día), mientras que el de alumnos que puntuaron a SRec con entre 4 y 4,5 puntos roza el 32% (50% el primer día de la sesión). De esta manera, el 50% de los estudiantes otorgó a SRec una nota media de satisfacción personal igual o superior a 4, frente al 82% del primer día.

El grupo del rango [3.5-4.0) supuso exactamente el 36,36% de los estudiantes, mientras el siguiente grupo, que evaluó a SRec con medias en el rango [3.0-3.5), supone el 9%. El grupo del rango [2.5-3.0) no obtuvo ningún representante. Tan sólo un estudiante otorgó una media personal por debajo del 2.5, el cual supone el 4,5% (puntuó a todas las preguntas sin distinción con un 2, lo cual hace dudar de su interés en rellenar el formulario).

Estos datos reflejan una supremacía del grupo que puntuó a SRec entre 4 y 5 (50%) frente a los que lo puntuaron entre 3 y 4, que suponen el 45,4%. Por otro lado, sumando los grupos mayoritarios, obtenemos que el rango [3.5-4.5) representa más del

68% de los encuestados, más de dos tercios. Tras él, el segundo grupo en importancia es el de mayor puntuación, que alberga al 13,6% de los estudiantes. Por el contrario, el 13% de los estudiantes dieron una nota inferior al 3,5, lo que mejora levemente el resultado de la tercera sesión (14%).

No todos los estudiantes respondieron a todas las cuestiones con una valoración numérica, de tal forma que su media personal se obtuvo teniendo en cuenta sólo las respuestas numéricas dadas.

6.8.2 Críticas y Sugerencias Recibidas en la Primera Sesión

Respecto a lo que más echaron en falta a la hora de trabajar con SRec, la información suministrada por los estudiantes se puede resumir en tres puntos principales:

- Algunos estudiantes criticaron el editor de código que proporciona SRec por no ajustarse a lo que ellos consideran un editor “ideal”. Esto se traduce en que éste debería estar integrado en la ventana principal, debería permitir automatizar más el trabajo de recargar clases editadas, debería permitir crear clases nuevas y no limitarse a editar clases ya existentes, debería ofrecer más funcionalidades como el coloreado de palabras reservadas y otras facilidades que proporcionan los editores... Además, cuando SRec detectase un error de compilación (lo cual puede darse al cargar una clase que contenga errores), debería dar acceso directo al editor de código, para facilitar su corrección por parte del usuario. Así, el editor de código centró las críticas durante el primer día de la sesión ya que los ejercicios pedían por primera vez implementar algoritmos desde cero.
- Hubo quien solicitó un depurador y que se marcaran en la vista de código las líneas de código que se van ejecutando. Este enfoque no se ajusta al funcionamiento del programa ni de sus animaciones, ya que éstas tienen en las subllamadas su paso natural de transición, no se basan en instrucciones simples de código, como ya se ha mencionado anteriormente.
- Algunas facilidades de instalación también fueron reivindicadas, sobre todo en lo referido a la necesidad que tenía SRec de que el usuario le indicara dónde se encuentra la máquina virtual de Java que se debe usar.

- La capacidad de mantener cargadas varias clases al mismo tiempo, de tal forma que se puedan ejecutar algoritmos que se encuentran en diferentes clases también fue requerida.

Como aspectos más positivos de SRec, destacan dos sobre todos los demás:

- Dos tercios de los alumnos indicaron, de una u otra manera, que la aplicación es capaz de ayudar a entender los algoritmos recursivos, resultando ser así una herramienta útil y que cumple el objetivo primario que dio pie a su implementación. Son varios los estudiantes que resaltan la capacidad de transmisión de conocimientos gracias a la representación gráfica que proporciona el programa, siendo el árbol de activación uno de los elementos mejor valorados de SRec. También fue remarcada la capacidad para dejar ver fácilmente los errores que se producen durante la ejecución del algoritmo cuando éste está mal programado.
- Una cuarta parte de los estudiantes resaltaron especialmente que SRec es muy fácil de usar. Así, se cumple uno de los requisitos que se han mantenido en pie durante todo el proceso de desarrollo de la aplicación: el aprendizaje de la herramienta debe llevar muy poco esfuerzo, y para ello es vital que su manejo sea sencillo e intuitivo.

Otros aspectos que contaron con el visto bueno de los estudiantes fueron la posibilidad de poder exportar el contenido de las vistas a animaciones en formato estándar GIF o realizar capturas estáticas en formatos PNG, GIF o JPG junto al hecho de que fuese considerado intuitivo y sencillo, viendo esto último como una cualidad. Además, se encontró entre las respuestas la sugerencia de que SRec debería emplearse en primer curso de la carrera a la hora de enseñar programación recursiva, por su alta capacidad de asistencia en el aprendizaje de esta compleja técnica.

Respecto a los aspectos negativos de SRec, las opiniones resultaron más heterogéneas, con muy poco respaldo cada una. Esto podría significar que son varios los puntos de SRec que podrían ser mejorados o modificados, pero que su importancia para los alumnos de forma global está muy limitada. Haciendo un repaso de los aspectos negativos encontramos el siguiente listado:

- Demasiadas opciones de formato: así, los cuadros de diálogo referidos a este campo podrían estar demasiado sobrecargados.

- Laborioso el proceso de cargar clases: los usuarios se enfrentan a un cuadro de diálogo no demasiado intuitivo durante el proceso de carga, lo que hace más pesado todo el proceso.
- Problemas de interfaz: que provocan que a algún estudiante le resultara la aplicación difícil de usar, o le pareciera que el programa cuenta con una interfaz “mediocre”, resultando complejo a veces encontrar la funcionalidad requerida.
- Problemas con árboles grandes: persisten algunas dificultades para trabajar con árboles de gran tamaño a pesar de haber introducido una vista global interactiva ya que si se amplía el espacio del árbol que se visualiza en la ventana, los datos contenidos en los nodos pueden volverse demasiado pequeños para una cómoda visualización.
- Problemas con ejecuciones largas: si la ejecución del algoritmo es larga y genera muchas subllamadas, esta ejecución y la posterior creación de las vistas puede suponer el trabajo intensivo de SRec durante varios segundos, tiempo que hace que SRec pueda ser considerado “lento”. Además, las limitaciones que impone Java respecto al uso de memoria pueden tener como resultado que las ejecuciones se queden sin memoria disponible asignada por Java, por lo que el algoritmo levanta una excepción y su visualización no puede mostrarse.
- La necesidad de seleccionar la máquina virtual de Java antes del primer uso de SRec también generó alguna mención negativa.
- El proceso de exportación de capturas recibió una crítica: debería permitir capturar las vistas completas y no sólo la parte visible en cada momento.
- De nuevo, el aspecto más criticado fue el editor de código. Sin duda alguna, el grueso del trabajo ha estado enfocado en las vistas y las diferentes funcionalidades del programa relacionadas con ellas, dejando de lado la edición de código que en esta sesión ha tenido como consecuencia una crítica casi generalizada (sí mayoritaria) acerca de esta parte del programa.

Por tanto, como conclusión, parece claro que SRec tiene varios pilares muy fuertes, como la facilidad de uso y la verdadera capacidad de asistencia en el aprendizaje de algoritmos recursivos, lo cual está respaldado por mayorías, y que, en

contra, existe una serie de aspectos negativos que son más débiles por contar con poco respaldo del alumnado en general pero que no por ello deben ser ignorados.

6.8.3 Críticas y Sugerencias Recibidas en la Segunda Sesión

En este segundo día de la sesión se introdujeron por parte de los estudiantes menos respuestas y de menor precisión, aunque igualmente son válidas para poder averiguar qué partes gozan de mayor aceptación y cuáles provocan mayor rechazo, confusión o problemas para emplear la herramienta. Además, se observan reivindicaciones, quejas y críticas muy similares a las del primer día.

Entre las partes más difíciles de usar aparece de manera destacada el cuadro de diálogo que aparece en la herramienta al cargar clases Java. Este cuadro de diálogo solicita al usuario que señale qué métodos contienen algoritmos diseñados bajo la técnica de “divide y vencerás”, requiriendo posteriormente al usuario indicar qué parámetros albergan la estructura de datos manejada por el algoritmo y los índices que delimitan qué parte de la estructura es manejada en cada subllamada.

También se hizo notar por parte del alumnado que la aplicación no es del todo intuitiva, no sólo por el cuadro de diálogo anteriormente citado, sino también por los iconos y nombres de algunas opciones, que en algunos casos no resultan explicativos sobre la funcionalidad que representan.

Tampoco resultaron fáciles de usar los cuadros de diálogo que permiten modificar la cantidad de información mostrada en pantalla. Además, hubo quien se quejó de que la ventana de la aplicación abre demasiados paneles, cada uno con su barra de desplazamiento, que puede llegar a despistar sobre cuántas y qué vistas se tienen en pantalla en cada momento.

En el momento de reivindicar características útiles, las carencias encontradas sobre la edición de código volvieron a salir a relucir, aunque también se pedía la posibilidad de compilar código “a nivel de métodos”, quizá intentando abstraerse del concepto de clase para centrarse más en el de método o algoritmo.

También se repitió una petición de la primera sesión, que es que SRec pueda capturar en archivos gráficos las representaciones enteras de las vistas y no sólo la parte visible en la ventana en el momento de efectuar la captura, si bien es, como ya se ha comentado, una limitación dada por la librería Swing lenguaje en el que está

programada la aplicación, Java (no obstante, esta dificultad pudo ser esquivada posteriormente gracias a la librería gráfica JGraph). Que SRec pueda autoconfigurar la máquina virtual de Java en su primer uso y que sea compatible con otras plataformas como MacOS o Linux son otras sugerencias que quedaron recogidas en los cuestionarios, junto con que la vista del árbol de activación realice un desplazamiento automático sobre la parte relevante del árbol a medida que se va visualizando la animación automatizada.

Como característica poco útil únicamente se mencionó el control de formato, dando a entender que es un cuadro de configuración complejo para una cuestión secundaria como es el coloreado de las vistas.

Aprovechando que esta sesión de evaluación se iba a celebrar en dos sesiones separadas por un periodo de casi un mes, se preguntó a los alumnos si habían empleado SRec y con qué fin entre el primer y segundo día de la sesión. Seis de los 27 (es decir, un 22%) sí emplearon SRec por su propia iniciativa, sin ninguna recomendación u obligación, para diversos fines que listamos a continuación:

- Algoritmos voraces (tres de los seis alumnos).
- Para comprobar algún ejercicio visto en clase y comprenderlo mejor (dos).
- Ver si ciertos algoritmos son eficientes (uno).

6.8.4 Conclusiones

Las principales conclusiones que se pueden extraer a nivel de usabilidad es que las nuevas funcionalidades (vistas para la técnica “divide y vencerás” y procesamiento de clases para tal fin) necesitaban una revisión para adecuarse más a los gustos y manera de proceder de los alumnos, con el fin de facilitarles la realización de las tareas.

Suponen estos resultados la constatación de lo sospechado en la anterior evaluación: el hecho de contar con mayor número de funcionalidades es todo un factor de riesgo contra la estabilidad de los niveles de simplicidad y usabilidad.

No obstante, resultó interesante apreciar a través del cuestionario de la primera sesión que las funcionalidades tradicionales habían aumentado su aceptación, por lo que las mejoras introducidas desde la anterior evaluación tuvieron un palpable efecto positivo que ahora había que buscar en las nuevas funcionalidades incorporadas ayudándonos de la información suministrada por los alumnos.

6.9 Evaluación de Usabilidad nº 5

Esta evaluación se celebró durante tres días, en las que los participantes eran alumnos de la asignatura obligatoria "Diseño y análisis de algoritmos" de 3º curso de Ingeniería Informática, impartida por los profesores Emilio José San Martín Fuentes, J. Ángel Velázquez Iturbide y Antonio Pérez Carrasco. Estas sesiones tuvieron lugar en el aula habitual de laboratorios empleando un ordenador por persona, en el horario habitual de la asignatura. La primera cita de la sesión fue el 7 de octubre de 2010, la segunda el día siguiente, 8 de octubre, mientras que la tercera sesión tuvo lugar el 15 de octubre de 2010.

Las sesiones siguieron, en términos generales, el mismo esquema de actuación que en evaluaciones anteriores, tanto por parte del profesorado como de los alumnos, si bien aumentó significativamente la cantidad de información recogida. La duración de cada cita fue de dos horas.

En cada una de ellas se planteó una práctica cuya complejidad iba aumentando a lo largo de los días. La primera sesión se limitaba a tareas de familiarización con la herramienta y a un ejercicio de baja complejidad, durante la segunda sesión se les pidió depurar un algoritmo de ordenación mientras que en la tercera sesión tuvieron que diseñar, implementar y depurar un algoritmo, dadas las especificaciones del problema.

Al principio de cada sesión siempre hubo una pequeña demostración del profesor para explicar algunos aspectos de la aplicación que posteriormente utilizarían los alumnos durante ese día. Tras ella, los alumnos comenzaban a afrontar individualmente o por parejas la correspondiente práctica propuesta.

Por último, el tercer día se les facilitó un breve cuestionario que pedía su valoración en una escala de 1 a 5 sobre ciertos aspectos concretos de la aplicación, permitiéndoles así calificar diferentes partes de la aplicación para valorar su calidad y facilidad de uso. Además, el cuestionario planteaba preguntas de respuesta abierta, de tal forma que los estudiantes podían expresar sus impresiones, críticas, sugerencias, dificultades encontradas, etc.

En esta ocasión se propuso explícitamente a los alumnos a que se identificaran para poder relacionar los trabajos realizados con las respuestas del cuestionario con el único ánimo de poder estudiar mejor la usabilidad de la herramienta. Esto se hizo con el propósito de contar con datos que permitan profundizar en las respuestas dadas

(haciendo entrevistas si se llegara a considerar necesario) y establecer relaciones entre sus impresiones expresadas en el cuestionario, los resultados de las prácticas y el uso de la aplicación registrado en sus archivos LOG.

Durante las distintas sesiones se llevaron a cabo por primera vez observaciones de comportamiento de los participantes. Hubo dos personas en el aula que se encargaron de anotar las diferentes preguntas que asaltaban a los alumnos, tanto sobre la aplicación como sobre el enunciado y la forma de resolver los problemas planteados. También se registró qué uso se hacía del papel como herramienta de visualización y cálculos de borrador y si se empleaba Internet para buscar más información sobre los algoritmos que entraban en juego en los ejercicios. Esta información se almacenó para intentar catalogar en perfiles a los alumnos y estudiar la relación con el tipo de uso que hicieron de la aplicación.

Tras esta experiencia, quedó constancia de que es materialmente imposible registrar todos los hechos relevantes que producen casi una cincuentena de personas que utilizan una aplicación software con sólo dos observadores. Por ello, deben entenderse los datos aportados en este apartado como orientativos.

También se registró la actividad de los alumnos mediante la capacidad de la aplicación de guardar automáticamente en un fichero cada acción del usuario, tomando nota de la fecha y la hora. Estos ficheros de actividad permiten cuantificar las visualizaciones que realizaron los alumnos para resolver cada problema, cuánto tiempo estuvieron empleando cada visualización, si navegaron erráticamente por los menús hasta encontrar la opción deseada, etc.

	Sesión 1	**Sesión 2**	**Sesión 3**
Informe de prácticas	X	X	X
Anotaciones de observaciones	X	X	X
Archivo LOG		X	X
Cuestionario			X

Tabla 28. Productos obtenidos en las sesiones de la quinta evaluación.

Tras el trabajo realizado, se tiene la información necesaria para poder asociar los informes de prácticas, el cuestionario, ficheros de registro de actividad y las anotaciones de las observaciones sobre cada alumno, por lo que se pueden establecer estudios profundos sobre el comportamiento de los participantes y el uso dado a la aplicación. Los ficheros de registro de actividad no han sido aún examinados en profundidad por lo que el estudio queda propuesto para abrir una nueva línea de investigación futura.

Dado que de esta evaluación se obtuvo una ingente cantidad de información de cada participante, se reúne en la Tabla 28 qué productos fueron cosechados en cada una de las tres sesiones de trabajo en el laboratorio.

6.9.1 Valoraciones Numéricas

Se adjunta en la Tabla 29 la puntuación que dieron 49 alumnos a SRec sobre ciertos aspectos generales (aportando la media y la desviación típica) a través del cuestionario suministrado en la tercera de las tres sesiones.

Cuestión	**Media**	**Desv. T.**
SRec me ha ayudado a ilustrar la definición inductiva del algoritmo	4,10	0,65
SRec me ha ayudado a ilustrar el comportamiento detallado del algoritmo	4,00	0,76
Calidad general de SRec para ilustrar algoritmos de Divide y Vencerás	3,98	0,71
SRec es fácil de usar	3,94	0,77
SRec me ha gustado	3,84	0,55

Tabla 29. Puntuaciones de aspectos globales en la quinta evaluación.

Estos resultados suponen una ligera mejora frente a los datos del segundo día de la cuarta evaluación de usabilidad, último cuestionario del que se tienen datos. La pregunta “SRec es fácil de usar” gana cuatro centésimas, las preguntas segunda y tercera sí obtienen mayores puntuaciones en alrededor de 3 décimas (4,10 y 4,00 frente a 3,83 y 3,68), la cuarta pregunta gana, al igual que la primera, únicamente 4 centésimas mientras que la última pregunta, “SRec me ha gustado” pierde una centésima.

La desviación típica de cada pregunta presenta un valor inferior al de la cuarta evaluación, lo que se traduce en que la variabilidad de los datos es menor, por lo que las respuestas están, al menos en su mayoría, más cerca de la media presentada, representando un mayor consenso a la hora de aportar estas puntuaciones a la aplicación, siendo por tanto, más generalizada la mayor aceptación de SRec.

En esta quinta evaluación, tres cuartas partes de los encuestados (75,51%) dieron una nota a SRec de 4 ó 5 a la hora de evaluar si SRec es fácil de usar (un 53,06% optó por el 4 y un 22,45% por el 5). Uno de cada cinco (20,41%) otorgó una puntuación de 3 mientras que tan sólo dos personas optaron por puntuar con un 2 tal cuestión. Por otra parte, SRec gustó considerablemente a cuatro de cada cinco alumnos, concretamente al 79,59%, que optaron por dar una puntación en la pregunta (e) de 4 ó 5 (73,47% para el

4, 6,12% para el 5). El 18,37% optaron por una posición intermedia mientras que tan sólo una persona puntuó con 2 esta pregunta.

El 81,63% de los encuestados puntuó con 4 ó 5 la calidad de SRec para ilustrar algoritmos diseñados bajo la técnica de Divide y Vencerás, un porcentaje que supera al de la evaluación cuarta de 2009. El 14,29% optó por una posición intermedia mientras que sólo dos personas (4,08%) dieron un 2 a SRec en la cuarta pregunta.

SRec supo ayudar a los alumnos en sus tareas durante las diferentes sesiones y así lo expresaron en el cuestionario. Alrededor de una cuarta parte de ellos dio un 5 (nota máxima) a SRec en las preguntas sobre si SRec les había ayudado (preguntas 2 y 3). Concretamente fueron un 26,53% para la pregunta 2 y un 24,49% para la pregunta 3. Más de la mitad se decantó por puntuar con un 4 tales preguntas (57,14% para la segunda y 55,10% para la tercera). Así, estas preguntas lograron que el 83,67% y el 79,59% de los alumnos dieran una respuesta de 4 ó 5 frente a los que optaron por la respuesta neutra (16,33% en ambas cuestiones). En la tercera pregunta dos personas decidieron dar un 2 a SRec. Se presenta en la Tabla 30 un resumen de las preguntas sobre aspectos específicos.

Cuestión	**C**	**DTC**	**FU**	**DTFU**
Estructura del menú principal	3,78	0,79	4,00	0,73
Iconos	3,59	0,83	3,49	0,99
Vista de árbol de activación	3,82	0,75	4,00	0,64
Visor de árboles grandes	2,82	0,85	3,27	0,94
Vista cronológica de la estructura	3,92	0,80	3,86	0,83
Vista de la estructura de datos	3,63	0,77	3,63	0,66
Control de zoom	3,24	0,92	3,33	0,96
Control de información que se muestra	3,35	0,66	3,51	0,70
Configuración de formatos gráficos	3,57	0,88	3,49	0,86
Controles de animación	3,96	0,78	3,96	0,81
Interacción con los paneles (mover, scroll, zoom…)	3,57	0,83	3,55	0,81
Proceso de generación de una animación	4,22	0,68	4,20	0,73
Proceso de almacenar/cargar una animación	3,73	0,88	3,71	0,88
Visualización almacenada en un fichero de captura	3,57	0,86	3,61	0,83
Media de las puntuaciones anteriores	3,62		3,68	

Tabla 30. Puntuaciones de aspectos concretos en la quinta evaluación.

El aspecto mejor valorado respecto a la calidad es el proceso de generación de una animación, seguido de los controles de animación, que también consiguieron ser la segunda característica mejor valorada en la anterior evaluación. Las puntuaciones más altas son prácticamente iguales respecto a la facilidad de uso.

La pregunta donde más variabilidad hubo en las respuestas fue la del control de zoom, que a su vez consiguió las segundas puntuaciones más bajas, por detrás del visor de árboles grandes.

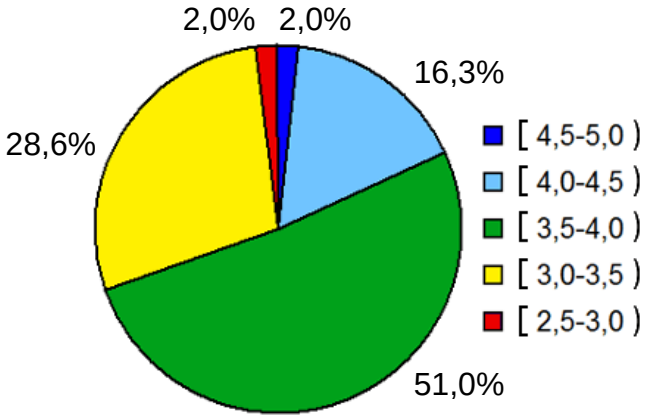


Ilustración 51. Aceptación personal de SRec (5ª evaluación)

Revisando, también a través de los cuestionarios, la aceptación personal de SRec (media individual de las notas numéricas suministradas para todas las cuestiones), se puede apreciar que se alcanza el 3.70 frente al 3.98 de la cuarta evaluación. Se presenta en la Ilustración 51 la distribución de las medias personales de aceptación.

El 18,36% de los alumnos otorgaron una media global a SRec entre 4 y 5 mientras que el grupo mayoritario está en el intervalo [3.5-4.0), que supone el 51,02% de los encuestados. Numeroso resulta también el grupo de estudiantes que se quedó en el intervalo [3.0-3.5), que representan el 28,57%.

Los resultados empeoran los obtenidos en la cuarta sesión, donde el 50% de los alumnos dio una nota igual o superior a 4. La diferencia en la media de aceptación personal entre repetidores de la asignatura (segunda matrícula o posterior) y los que cursan la asignatura por primera vez es muy pequeña, 3,64 frente a 3,75.

6.9.2 Críticas y Sugerencias Recibidas

En las diferentes respuestas abiertas del cuestionario los alumnos expresaron algunas ideas positivas acerca de SRec. Quedan recopiladas a continuación.

La idea de que SRec es fácil de usar, con elementos de interfaz sencillos resultando por tanto intuitivo en su uso, está secundada por nueve personas como respuesta a la pregunta “Las partes que te parecen más difíciles de usar (si las hay) son:”. De tal forma, ante una pregunta que intenta encontrar puntos negros de SRec, el

18,36% ofrece una respuesta positiva, mientras que 24 alumnos (48,97%) no referenciaron ninguna parte de SRec como especialmente difícil de usar.

Otra de las preguntas abiertas del cuestionario era “Di qué características te parece que podrían ser útiles pero SRec carece de ellas”. El número de personas que expresaron no echar nada en falta fue de 4 (8,16%) mientras que fueron 10 (20,41%) las que dejaron la pregunta en blanco, por lo que casi 3 de cada 10 alumnos no aportaron ninguna sugerencia para incorporar alguna funcionalidad o característica a la aplicación.

Fueron trece personas (26,53%) las que ante la pregunta “Di qué características de SRec te parecen tan poco útiles que las suprimirías” dijeron que no encontraron funcionalidades o partes inútiles, por lo que no suprimirían nada. Por su parte, 19 personas dejaron la pregunta en blanco sin aportación útil (38,77%). Así las cosas, sólo un tercio de los alumnos encontró funcionalidades poco o nada útiles en la aplicación.

Por último, el 20,41% de los alumnos encuestados (10) utilizaron espontáneamente y por su propia cuenta SRec para estudiar o repasar contenidos de la asignatura. Algunos de ellos lo utilizaron para estudiar el uso del algoritmo de ordenación Mergesort, mientras que otros destacan como justificación de su uso las facilidades que da la herramienta para ver el funcionamiento del algoritmo paso a paso de una manera visual.

Estas buenas valoraciones se unen a las recogidas en los informes de prácticas donde, la mayoría de los alumnos comentaba espontáneamente (sin que se les pidiera) en qué medida SRec les parecía útil, intuitivo o les había ayudado a realizar la práctica. Así, los informes recogen afirmaciones como “cumple perfectamente sus funciones”, “nos facilita bastante el trabajo”, “es una gran ventaja el contar con las diferentes vistas ofrecidas”, “muy intuitivo y de rápido aprendizaje”, “me ha servido para comprender mejor”, “es una buena herramienta para la simulación de programas recursivos” o “total control sobre la animación”. Algunos alumnos destacaron la facilidad para hacer capturas y para editar código, “la posibilidad de ver con qué parte del vector trabajas”, y resaltaron, ante todo, la capacidad expresiva de las vistas.

Se repasan ahora las respuestas dadas que suponen valoraciones negativas de SRec, aportando sugerencias, alternativas o ideas no desarrolladas. Ante la pregunta “Las partes que te parecen más difíciles de usar (si las hay) son”, las respuestas dadas fueron las siguientes:

- Dificultad para recordar/saber cómo introducir la información necesaria (parámetros) para ejecutar un algoritmo por ser poco intuitiva. Seis personas respaldaron esta opinión.
- Dificultad inicial para saber qué significan los iconos y elementos de la interfaz, al principio el uso es poco intuitivo. Tres personas dieron respuestas en esta línea.
- La edición de código. Dos personas fueron las que situaron en el punto de mira el editor de código.
- Algunas opciones no están identificadas de manera intuitiva en los menús.
- Dificultad para comprender los valores que deben introducirse para habilitar las vistas de la técnica Divide y Vencerás.
- Costoso entender la vista cronológica.
- Complicado manejo de árboles grandes.
- Limitación de SRec en cuanto al número de clases Java que es capaz de manejar en un instante de tiempo (sólo una).

El proceso de lanzar un método para ver su ejecución parece ser uno de los puntos más conflictivos del programa y que más confusión crea. El cuadro de diálogo es de gran tamaño y aparece la signatura de los métodos disponibles para lanzar que están implementados en la clase cargada. Se deberán estudiar por tanto modificaciones al respecto para mejorar el proceso de cargado de clase y de lanzamiento de visualizaciones.

También fueron varias las personas que resaltaron que inicialmente la aplicación no resulta intuitiva; esto deja ver que el proceso de aprendizaje de la herramienta puede ser necesario pero no es posible determinar por ello si este proceso de aprendizaje es costoso o no para los usuarios. También se resaltó que los nombres de las opciones son poco intuitivos.

La edición de código, ahora integrada en la ventana principal, también registró alguna sugerencia de mejora, no en vano fue un aspecto que recibió menciones negativas tanto en estas como en otras preguntas abiertas del cuestionario. En la evaluación anterior se pidió la integración de la edición de código en la ventana principal y, una vez implementada, lo que se pide ahora son mejoras específicas, como

ya se verá en otras preguntas abiertas. Al ser una de las funcionalidades más nuevas es lógico que sea también una de las más susceptibles a recibir críticas, pues el resto de funcionalidades se encuentran ya en un estado mucho más refinado.

La segunda de las preguntas abiertas planteadas en el cuestionario, “Di qué características te parece que podrían ser útiles pero SRec carece de ellas”, dejó ver qué echaron en falta los alumnos durante la utilización de la herramienta en las prácticas:

- Mejor editor de código (más posibilidades como señalar errores, autocompletado, opciones de cortar, copiar y pegar, opción deshacer, opciones de configuración, tabulador multilínea…). Doce personas se refirieron al editor de código para sugerir mejoras.
- Portabilidad entre plataformas (Linux, MacOS…). Fueron seis las personas que demandaron esta compatibilidad.
- Poder visualizar el valor de variables locales en cada momento. Cinco personas pidieron incorporar esta funcionalidad de depuración.
- Que avise de errores de ejecución (e indique, por ejemplo, la línea de código donde se produce). Tres personas mencionaron esta solicitud tras programar erróneamente su algoritmo con SRec sin poder utilizar la herramienta para identificar el error de ejecución.
- Captura de la vista de traza. Dos personas echaron en falta no poder exportar en formato gráfico esta vista pese a que en realidad SRec da la opción de exportarla en formato HTML, de tal forma que se pueda utilizar su contenido textual.
- Señalar en el código la rama que se está desplegando. Dos personas indicaron que podría ser interesante ver resaltada la línea de código que se ejecuta al abrir cada nueva subllamada.
- Superar la dificultad de SRec de no poder emplear archivos Java en directorios cuyo nombre tiene caracteres acentuados.
- Existencia de un manual de uso (existe, si bien no se les facilita para poder recoger las dudas directamente mediante las preguntas que realicen).

- Distintos métodos abreviados de teclado, ya que los actuales fueron criticados por coincidir con funcionalidades diferentes en otros programas de uso común.
- Poder establecer un punto de inicio y otro de final para la animación automática.
- Mejoras en la visualización de árboles grandes (sin concretar qué aspectos: calidad de visión, facilidad de navegación, o cualquier otro posible aspecto).

Como se puede ver, fueron doce personas (una cuarta parte de las que contestaron el cuestionario) las que mencionaron algún aspecto relacionado con el editor integrado añadido para esta nueva evaluación de SRec. Se pidió muy insistentemente la posibilidad de deshacer cambios realizados inmediatamente antes, solicitud acompañada de posibilidades de autocompletado y de otras menos requeridas como mejoras en la tabulación, facilidad de copiar y pegar con botones, etc.

Seis personas, lo que supone uno de cada ocho alumnos, echaron en falta poder ejecutar SRec en sus equipos MacOS o Linux al verse obligados a ejecutarlo en los equipos Windows de los laboratorios, que no ofrecen un rendimiento óptimo debido a su configuración, condición que también pudo empeorar colateralmente la imagen de SRec.

Por otro lado, cinco personas mencionaron la necesidad y utilidad de poder ver el contenido de las variables en cada paso que se da. Actualmente SRec sitúa cada paso de la visualización en la apertura o conclusión de una llamada recursiva, no avanza sentencia a sentencia, por lo que introducir esa posibilidad modificaría sensiblemente el concepto de la aplicación y la alejaría de su objetivo fundamental, la visualización de la recursión.

Sobre la petición de que SRec informe sobre los errores de ejecución, debe decirse que actualmente se informa del tipo de error que se produce (como por ejemplo, desbordamientos de pila, punteros a valores nulos, etc.) si bien no se indica la línea concreta de código donde se produce. Se deberá, por tanto, lograr que la aplicación pueda recuperar más información desde la Máquina Virtual de Java para poder suministrarla al usuario.

Por tanto, en esta pregunta se apreciaron carencias localizadas fundamentalmente en el recién estrenado editor de código, y en labores de depuración

(contenido de variables locales y localización de errores de ejecución) así como en aspectos técnicos (compatibilidad entre plataformas), pero no en las vistas ni en la capacidad de ilustración de los algoritmos.

La tercera de las preguntas abiertas que se encontraron los alumnos durante la realización el cuestionario fue “Di qué características de SRec te parecen tan poco útiles que las suprimirías”. Ésta recogió algunas de las opiniones ya vistas en la pregunta anterior. Los temas tratados en las respuestas son los siguientes:

- La vista de pila, pues proporciona información ya contenida en el árbol. Cuatro personas mencionaron esta vista.
- Cambiar los métodos abreviados de teclado, ya que los actuales coinciden con funcionalidades diferentes en otros programas de uso común. Fue sugerido por dos personas.
- La posibilidad de cambiar los colores de las vistas. Fue resaltado por dos personas.
- La vista de traza, no es fácil de ver.
- Los iconos tienen demasiado color, “no es necesaria tanta calidad”.
- La opción de seleccionar la cantidad de información mostrada (elegir entre ver sólo datos de entrada, datos de salida o ambos, quitar subárboles saltados, quitar o atenuar nodos históricos…).

En general, los alumnos se muestran reticentes a solicitar o sugerir la desaparición de una funcionalidad ya instalada, en algunas respuestas subyace la idea de “cuantas más funcionalidades tenga, mejor”. En ciertas ocasiones, sus comentarios se pueden traducir en sugerencias de mejoras, más que en críticas al uso. No obstante, algunos de ellos han afirmado que la vista de pila es totalmente prescindible, resultando para ellos más clara la vista del árbol, y que, existiendo esta, le deja a la vista de pila un papel de redundancia más que de información complementaria.

Se sugiere la retirada de algunos atajos de teclado y también se menciona como prescindible la opción de configurar el coloreado de las vistas.

6.9.3 Conclusiones

La quinta evaluación lleva a la conclusión de que los usuarios finales no tienen porqué ser capaces de diseñar adecuadamente un programa que ellos mismos vayan a usar. La prueba es que habiendo implementado en una manera fiel las sugerencias recibidas en la cuarta evaluación, como por ejemplo el editor integrado en la ventana principal de SRec, las puntuaciones de facilidad de uso y calidad han descendido ligeramente, en lugar de mostrar una evolución positiva, como sería de esperar. Precisamente el editor de código centra algunas de las sugerencias, nuevas respecto a la evaluación anterior.

Abre por tanto esta quinta y última evaluación un periodo de reflexión que lleva a este trabajo a buscar diversos objetivos concretos:

- Solución para la visualización de árboles grandes, pues es uno de los talones de Aquiles de la herramienta. Queda demostrado por obtener en todas las evaluaciones una de las peores puntuaciones.
- Mejor integración de las funcionalidades en la interfaz, mejorando los iconos, las definiciones de las opciones y la organización de los menús para hacer de la aplicación un sistema más intuitivo. También se puede unir una simplificación de la configuración tipográfica de SRec para culminar esta tarea.
- Mejorar el procesado de clases para evitar la existente carga sobre el usuario acerca de la elección de métodos para la habilitación de vistas específicas de la técnica “divide y vencerás”. A menudo es confundida la información que debe introducirse en el procesamiento de las clases y en el lanzamiento de ejecuciones de métodos.

Por tanto, deberá establecerse un camino de progreso que mejore las características existentes antes de incorporar nuevas funcionalidades.

6.10 Conclusiones del Capítulo

Este capítulo refleja el extenso trabajo realizado acerca de la evaluación de la usabilidad de la aplicación mediante cuestionarios. Ésta es una técnica de carácter eminentemente subjetivo pero que, con la contribución de un elevado número de

participantes, es capaz de reflejar aciertos y carencias reales de aquello que se evalúa, tanto referido a la usabilidad como a eficacia. Tras cinco evaluaciones, la aplicación ha tomado una forma afín a los profesores y, sobre todo, a los alumnos, que han ido sugiriendo funcionalidades, criticando la forma de trabajar con SRec y valorando diferentes aspectos y su imagen general, respecto a la calidad y la facilidad de uso.

Se ha repasado en este capítulo el método general, indicando el número de participantes en cada sesión, los productos obtenidos, las medias y desviaciones típicas en las valoraciones, así como un listado de sugerencias, críticas y puntos bien valorados de SRec.

Como ya se ha ido explicando a lo largo de las conclusiones de cada una de las sesiones, se ha constatado que un mayor rango de funcionalidades puede dañar el nivel de usabilidad, sobre todo si se incorporan procesos complejos y poco intuitivos. También se ha llegado a la conclusión de que seguir las directrices de los alumnos (tomados como individuos de los experimentos) no tiene porqué suponer un aumento de la usabilidad, pues en ocasiones se ha recibido un descenso en las puntuaciones por incorporar lo que se sugirió en una evaluación anterior. Queda en evidencia por tanto que el usuario de una aplicación a menudo no sabe qué es lo mejor para mejorar la usabilidad del programa que maneja, por lo que las sugerencias y críticas deben aceptarse, pero ejerciendo un análisis que determine la validez de la misma y su mejor modo de influencia en la aplicación, si es que procede.

Se han identificado varios problemas que deberán ser tratados para mejorar la usabilidad y utilidad de la aplicación. En cualquier caso, se ha logrado una aplicación muy satisfactoria desde todos los puntos de vista (funcionalidad, usabilidad, eficacia...).

Finalmente, la valoración global que se hace es positiva, por el aprendizaje cosechado, que servirá para mejorar más eficazmente SRec en el futuro, y por la completitud de SRec desde un punto de vista funcional. Los alumnos, como ya se ha explicado, decían espontáneamente en sus informes de prácticas cómo SRec les ayudó a realizar las prácticas, lo fácil de usar que les resultaba (a veces tras un comienzo algo complejo) y la alta capacidad de ilustración que tiene.

Capítulo 7. Evaluación de Usabilidad con otros Métodos

La usabilidad ha sido estudiada también a través de otros medios más allá de los cuestionarios, como los archivos de registro de actividad de la aplicación, lo cual permite realizar una evaluación objetiva de aspectos como la eficiencia. Gracias a ellos se ha estudiado el uso que han hecho de la aplicación los alumnos (cuántas veces cargaban clases, cuántas visualizaciones generaron, qué errores encontraron, qué tamaño solían dar a los datos de entrada, etc.). Estos primeros resultados obtenidos, de carácter provisional, están recopilados en el apartado 1 de este sexto capítulo.

Por otra parte, tal y como se comentó en el capítulo anterior, se realizaron ciertas observaciones sobre los alumnos y la manera de proceder durante el transcurso de la quinta evaluación de usabilidad de SRec. En el apartado 2 de este capítulo se exponen los principales hechos constatados extraídos desde esas observaciones, que conservan un enfoque subjetivo. Posteriormente en el apartado 3 se realizan unas breves reflexiones sobre otros tipos de evaluaciones de usabilidad posibles de realizar con la información de que se dispone.

7.1 Utilización de SRec por Parte de los Alumnos

La utilización de SRec por parte de los alumnos es una cuestión en la que tan sólo se ha realizado un estudio preliminar hasta el momento, recabando ciertos datos pero sin extraer conclusiones definitivas, siendo necesario para ello un estudio más exhaustivo y profundo que relacione esta información con la proporcionada por otros productos de los mismos usuarios.

Como ya se ha explicado, durante las sesiones 2 y 3 de la quinta evaluación de usabilidad de SRec, se propusieron dos prácticas evaluables. Se recogieron, entre otros productos de esas dos prácticas realizadas por los alumnos, un archivo por cada práctica de cada uno de ellos que recoge los principales sucesos acontecidos durante la utilización de la aplicación.

En esta navegación por ellos que aquí se presenta, se han centrado los esfuerzos en recoger los siguientes datos:

- Sesiones de trabajo: se cuentan y miden las sesiones de trabajo de cada alumno para conocer qué duración tenían, y obtener ciertas estadísticas.
- Errores: se contabilizaron los errores en total entre todas las sesiones de trabajo, así como la media y el número de errores de las primeras sesiones, la más propensa a “tropezarse” con la interfaz de la aplicación.
- Número de visualizaciones creadas satisfactoriamente, separándolas de aquellas cuyo proceso de creación lanzó algún error (implicara su aborto o no).
- Número de veces que se cargan las clases, lo que puede dar una idea de cómo fue el trabajo de depuración de los algoritmos y de cuántas veces necesitaron visualizar el algoritmo para poder dar con el código correcto.
- Número de exportaciones gráficas, que indica la capacidad de SRec para proporcionar una representación óptima para otros fines que no sean la visualización interactiva así como la facilidad de uso de esta característica.

Las tres primeras cuestiones están relacionadas con la usabilidad de la aplicación y la aparición y gestión de errores, mientras que las siguientes permiten acercarse al proceso de la resolución del problema por parte de los alumnos mediante la revisión del proceso de depuración (relacionado con el número de veces que se necesita cargar una clase hasta verificar que está bien programada) o del encuentro de un ejemplo válido representado (número de veces que se realizan exportaciones gráficas).

7.1.1 Sesiones de Trabajo

Se cuenta por sesión de trabajo cada vez que SRec es abierto, utilizado y cerrado, independientemente del tiempo transcurrido desde la anterior sesión (ya sean días, horas, minutos o segundos).

En ocasiones las sesiones acaban porque SRec no puede continuar ante algún error en el procesamiento de las clases al contener estas errores, si bien en otras ocasiones los alumnos cierran la aplicación libremente y vuelven a abrirla para realizar otras tareas o realizar la misma en una nueva ocasión si olvidaron finalizar alguna parte de los ejercicios propuestos.

Un mayor número de sesiones puede significar menor capacidad para realizar adecuadamente las tareas, bien por olvidos, bien por dificultad para realizar las tareas adecuadamente la primera vez que se llevan a cabo.

Se ofrecen a continuación en la Tabla 31 una serie de valores recogidos:

	Trabajo Día 2	**Trabajo Día 3**	**Global**
Número de registros LOG	41	30	58
Alumnos	51	48	-
Número de sesiones registradas	124	93	217
Número de sesiones bien cerradas	116 (94%)	60 (65%)	176 (85%)
Media de número de sesiones por trabajo	3,02	3,1	-
Media de duración de sesiones (min.)	76,07	39,11	62,27
Máximo número de sesiones por alumno	10	8	10

Tabla 31. Información sobre las mediciones realizadas.

El número global de registros LOG no es la suma de los archivos cosechados en cada una de las sesiones debido a que se han de filtrar repeticiones en los datos (lo que reduce el número necesario de ficheros que se deben analizar de manera global) y a que la composición de los grupos varió entre ambos días. También debe tenerse en cuenta que el grupo de alumnos que participó en el segundo día no fue exactamente el mismo que el del tercer día.

Tal y como refleja la tabla, el número medio de sesiones realizadas con SRec es prácticamente idéntico para los dos días, en torno a 3. No obstante, el número absoluto de sesiones registradas es mayor el día 2 porque también lo es el número de ficheros de registro (participantes) obtenidos.

Por otra parte, crece considerablemente el número de sesiones de SRec en las que éste tuvo algún problema y no pudo cerrarse correctamente, o bien el alumno lo cerró abruptamente en mitad de algún proceso (como la exportación de animaciones GIF, por ejemplo). Este aumento puede deberse al hecho de que los alumnos empezaron a programar un algoritmo desde cero y éste pudo generar algún problema en SRec o la máquina virtual de Java que impidiera a SRec recuperarse del mismo, siendo necesario su cierre. Los cálculos globales (sumando todas las sesiones registradas) indican que el 85% de las sesiones fueron cerradas por el usuario y no por la aparición de errores.

Un dato interesante es el valor medio de la duración de las sesiones, que se reduce considerablemente aun realizando tareas más complejas, lo que podría explicarse por un mayor dominio de la aplicación por parte de los alumnos.

Por último, y tal y como refleja la Tabla 31, el mayor número de sesiones necesitadas fue de 10 para realizar la tarea del día 2, mientras que fue de 8 para el día 3. No obstante, estas cifras no son especialmente significativas al poder existir sesiones de poca duración y sesiones de larga duración, donde la cantidad de trabajo productivo realizado puede variar considerablemente.

7.1.2 Procesamiento de Clases y Uso de Visualizaciones

Como ya es conocido, para poder obtener una visualización con SRec, primero es necesario cargar (y procesar) una clase Java y posteriormente lanzar la ejecución del método que contiene el algoritmo que se desea visualizar.

La primera parte de este proceso puede dar lugar a errores de valores (cuando se quiere activar para algún método las vistas de la técnica “divide y vencerás” y se introducen mal los números de los parámetros que delimitan la parte de la estructura que se maneja en cada subllamada recursiva) y a errores del sistema de ficheros (cuando SRec no tiene permisos de escritura y, por tanto, no puede generar código alternativo ni compilar tal código, situación comentada en el apartado “3.1.1 Limitaciones de Uso“).

La segunda parte de este proceso tiene un carácter más imprevisible, pues los errores y excepciones levantadas dependen del código que escriba el alumno, siendo el desbordamiento de pila, la ejecución infinita y la insuficiencia de memoria los más frecuentes.

La Tabla 32 refleja el número de cargas y procesamientos de clases que se realizaron en total, así como aquellos que levantaron al menos un error de valores o relacionados con el sistema de ficheros.

	Trabajo Día 2	**Trabajo Día 3**	**Global**
Número de procesamientos totales	483	565	1048
Número de procesamientos que levantaron al menos un error de valores	10 (2,07%)	11 (1,95%)	21 (2,00%)
Número de procesamientos que levantaron al menos un error de ficheros	13 (2,69%)	4 (0,71%)	17 (1,62%)

Tabla 32. Número de procesamientos de clases Java.

Se puede apreciar que la proporción de errores decrece una vez que los alumnos van conociendo la manera exacta en que se introducen esos valores y configuran SRec de manera adecuada para que pueda almacenar información en el sistema de ficheros.

Por otra parte, la Tabla 33 se centra en la segunda parte del proceso, el lanzamiento de las ejecuciones, indicando además el número de exportaciones gráficas realizadas con SRec para la inclusión de las mismas en los informes de prácticas que debían presentar.

	Trabajo Día 2	**Trabajo Día 3**	**Global**
Número de lanzamientos de métodos	650	370	1020
Número de lanzamientos de métodos que resultaron satisfactorios	556 (85,54%)	246 (66,49%)	802 (78,63%)
Número de exportaciones totales	177	93	270
Número de exportaciones por cada visualización abierta	0,31	0,38	0,34

Tabla 33. Lanzamientos de métodos y exportaciones a formatos gráficos.

Tal y como se puede ver, las incidencias en las ejecuciones de métodos crecieron debido a que en el día 3 lo que se ejecutaba eran algoritmos programados por los propios estudiantes, que necesitaban habitualmente de un proceso de depuración para eliminar bucles infinitos, un uso inapropiado de la memoria, etc. Aún así, sólo resultaron problemáticas un tercio de las ejecuciones de los algoritmos en total.

Los alumnos realizaron un número similar de exportaciones a ficheros gráficos en ambos días, aproximadamente una exportación por cada tres algoritmos visualizados. Un número alto de visualizaciones habría indicado la dificultad de encontrar una exportación satisfactoria para su posterior inclusión en el informe de prácticas. Es complicado, no obstante, determinar un umbral que separe los valores “grandes” de los “pequeños” respecto al número de exportaciones, pues depende de lo que se les pida en el informe de prácticas pero también del número de veces que lancen los algoritmos mientras depuran.

Por último, se han medido también los tamaños más usados de los vectores en aquellos métodos que requerían la inserción de este tipo de estructuras de datos de una dimensión. Se detallan estos valores en la Tabla 34.

Sorprende el alto uso de tamaños que no son potencia de 2, como los tamaños de 5, 6, 7 y 9, sobre todo. Era de esperar que los alumnos utilizasen vectores de tamaños que son potencia de 2 (fundamentalmente 4 y 8) al ser más intuitiva la partición de los

vectores en mitades para continuar el proceso recursivo que permita alcanzar los casos base. Sí se pudo constatar que en ocasiones se realizaban a propósito ejecuciones con vectores de tamaño impar para asegurarse de la corrección del algoritmo escrito, y de que éste funciona con todo tipo de vectores.

	Trabajo Día 2	**Trabajo Día 3**	**Global**
Número total de vectores usados	600	338	938
Número total de vectores de longitud 1	5 (0,83%)	12 (3,55%)	17 (1,81%)
Número total de vectores de longitud 2	12 (2,00%)	32 (9,47%)	44 (4,69%)
Número total de vectores de longitud 3	24 (4,00%)	62 (18,34%)	86 (9,17%)
Número total de vectores de longitud 4	159 (26,5%)	177 (52,37%)	336 (35,82%)
Número total de vectores de longitud 5	158 (26,33%)	31 (9,17%)	189 (20,15%)
Número total de vectores de longitud 6	107 (17,83%)	14 (4,14%)	121 (12,90%)
Número total de vectores de longitud 7	37 (6,17%)	3 (0,89%)	40 (4,26%)
Número total de vectores de longitud 8	19 (3,17%)	1 (0,30%)	20 (2,13%)
Número total de vectores de longitud 9	24 (4,00%)	1 (0,30%)	25 (2,67%)
Número total de vectores de longitud 10	33 (5,5%)	4 (1,18%)	37 (3,94%)
Número total de vectores de longitud 11 a 64	22 (3,77%)	1 (0,30%)	23 (2,45%)
Tamaño medio de vectores contabilizados	5,76	3,85	5,08

Tabla 34. Tamaño de los vectores usados por los alumnos.

En la sesión del día 3 los vectores de tamaño 4 duplicaron su proporción de uso, alcanzando el 52,37% de utilización, más que el resto de posibles longitudes juntas.

En el resultado global, los tamaños más usados fueron los de longitud 4 (el más usado en las dos sesiones), 5 (el segundo más usado el día 2), 6 y 3 (el segundo más usado el día 3). La media final es de 5,08 posiciones, un vector que puede resultar demasiado pequeño para ilustrar adecuadamente determinados algoritmos, como los de ordenación.

7.1.3 Errores durante el Uso de SRec

Uno de los aspectos más interesantes a la hora de evaluar la usabilidad de una aplicación es, sin duda, el número de errores que se cometen. No sólo conviene estudiar términos absolutos, también es necesario ver si el número de errores decrece gradualmente con el uso de la herramienta, si inicialmente la utilización de SRec está atropellada por un número demasiado elevado de errores, y de qué tipo y criticidad son estos errores, así como si el usuario tiene posibilidad y capacidad de reponerse ante ellos.

Con toda esa información debería poder establecer un conjunto de causas que ayuden a encontrar las soluciones a ese número de errores encontrado por los usuarios.

Se presentan a continuación en la Tabla 35 y en la Tabla 36 los totales, las medias, mínimos y máximos del número de errores realizados para las primeras 6 sesiones de trabajo con SRec de cada alumno para cada una de las tareas propuestas en los días 2 y 3. Se aportan sólo las seis primeras por ser las más representativas, ya que a partir de la séptima el número de errores es cero en prácticamente todas las sesiones, además de que no todos los alumnos necesitaron abrir siete sesiones de SRec (algunos incluso menos) para llevar a cabo las tareas.

	Trabajo Día 2	**Trabajo Día 3**
Total de todos los errores producidos	179	183
Media por trabajo	4,36	6,10
Media de todas las sesiones (sesiones/errores)	1,44	1,96
Media de 1ª sesión	1,90	3,77
Media de 2ª sesión	0,75	0,92
Media de 3ª sesión	1,70	1,88
Media de 4ª sesión	0,59	0,85
Media de 5ª sesión	0,25	0,16
Media de 6ª sesión	0,00	0,00

Tabla 35. Errores producidos en términos medios.

	Trabajo Día 2	**Trabajo Día 3**
Mínimo/Máximo de la 1ª sesión	0 / 9	0 / 17
Mínimo/Máximo de la 2ª sesión	0 / 4	0 / 4
Mínimo/Máximo de la 3ª sesión	0 / 7	0 / 15
Mínimo/Máximo de la 4ª sesión	0 / 3	0 / 3
Mínimo/Máximo de la 5ª sesión	0 / 3	0 / 1
Mínimo/Máximo de la 6ª sesión	0 / 0	0 / 0

Tabla 36. Errores producidos en términos mínimo y máximo.

Tal y como reflejan los datos, fue algo más problemático el tercer día que el segundo, debido fundamentalmente a la mayor complejidad de la tarea que tenían que realizar con SRec (depuración de un algoritmo dado el día 2 frente a la escritura y depuración de un algoritmo propio el día 3), y a que requería una mayor interacción con la misma (edición, compilación, realización de pruebas de resultado imprevisible…).

También se puede apreciar que las primeras tres ocasiones en que los alumnos emplean SRec encuentran más errores que en siguientes sesiones de trabajo, cuando ya conocen y se han acostumbrado a las características de la aplicación. Este hecho queda reflejado tanto en el número medio de errores por sesión como en el máximo número de errores de cada sesión con SRec. Salta a la vista la rotura de la tendencia de la tercera

sesión del día 3, cuyos datos (media y máximo) son fácilmente alterables con que uno o varios alumnos, por motivos cualesquiera, generaran un inusual número de errores.

A continuación se indican los tipos de error contabilizados, que se dividen en cuatro categorías: errores de parámetros, errores de especificación de visualización de “divide y vencerás”, errores de ejecución y errores del sistema de ficheros.

Los errores de parámetros se dan cuando el usuario intenta insertar los valores que debe emplear el método que ha seleccionado para ejecutar y no los escribe adecuadamente, bien porque no sigue la sintaxis adecuada (escribir entre llaves los arrays, por ejemplo) o porque el tipo no se ajusta al esperado por el método (introducción de un valor real cuando se espera un entero). SRec informa en ese caso mediante un cuadro de diálogo manteniendo la posibilidad de insertar un valor adecuado a continuación.

Los errores de especificación de visualización de “divide y vencerás” se dan cuando el usuario elige habilitar para ciertos métodos las vistas específicas de la técnica “divide y vencerás”, pues deberá indicarle a SRec qué parámetros son los que delimitan la parte de la estructura de datos manejada por el algoritmo en cada subllamada recursiva. Para muchos alumnos este cuadro de diálogo resulta confuso y no tienen claros los valores que deben insertar, de ahí que no lo rellenen adecuadamente y SRec se lo haga saber mediante un cuadro de diálogo de error.

Los errores de ejecución surgen cuando el algoritmo, mal programado, provoca una excepción (por ejemplo, desbordamiento de pila o insuficiencia de memoria física). La aplicación cesa entonces la ejecución, informa del error producido (y de la excepción Java levantada) y aborta la creación de la visualización. Se ha de tener en cuenta que estos errores se dan por un mal diseño o implementación de los algoritmos que se intentan visualizar, no de SRec.

Los errores en el sistema de ficheros aparecen en determinadas ocasiones cuando se trabaja en sistemas sin permiso de administración, en determinados dispositivos externos (discos duros o memorias conectables a través del puerto USB) o si la ruta del directorio contiene determinados caracteres especiales. En estos casos, la aplicación le indica al usuario que no puede escribir en disco, lo que resulta crítico para poder compilar clases y guardar contenidos de diferente tipo.

Otros errores, no contabilizados de manera específica, son, por ejemplo, los relacionados con la Máquina Virtual de Java y su configuración inicial. Si SRec no puede encontrarla por sí mismo, se lo indicará al usuario para que la instale manualmente.

La proporción en que estos se dieron durante la evaluación aparece reflejada en la Tabla 37. Tal y como refleja la tabla, los errores que dependen de la interfaz (inserción de valores y números de parámetros) decrecieron considerablemente de la sesión 2 a la sesión 3, pasando del 49% al 31% entre ambos tipos de errores. Por su parte, también bajó el número de errores en el sistema de ficheros así como el de otros tipos de errores, mientras que subió ligeramente el de otros tipos de errores no contabilizados.

	Día 2	**Día 3**	**Total**
Total de errores	179	183	362
Errores de parámetros	57 (32%)	32 (17%)	89 (25%)
Errores de especificación DYV	30 (17%)	25 (14%)	55 (15%)
Errores de ejecución	64 (36%)	112 (61%)	176 (49%)
Errores del sistema de ficheros	22 (12%)	4 (2%)	26 (7%)
Otros	6 (3%)	10 (5%)	16 (4%)

Tabla 37. Mediciones sobre los tipos de errores producidos.

Significativo, a la par que esperable, fue el aumento de errores de ejecución desde el día 2 hasta el 3. Esto vino motivado, lógicamente, por la mala calidad de las primeras versiones probadas de los códigos escritos por los alumnos, que contenían habitualmente bucles infinitos o sucesiones infinitas de llamadas recursivas que provocaban desbordamientos de pila. Este tipo de errores pasaron del 36% al 61%, suponiendo casi la mitad del total de errores que tuvieron lugar durante la medición realizada en términos globales. Fue el tipo más numeroso, seguido por los errores de parámetros, con la mitad de incidencias.

7.2 Observaciones Realizadas sobre los Usuarios

Los principales hechos relacionados con la usabilidad que se recogieron mediante estas observaciones fueron los siguientes:

- La proporción de gente que hizo uso de papel en al menos una de las sesiones para hacer anotaciones o bien representar grafos o trazas. Fueron en total 20 personas, lo que supone el 40.81%. El uso mayoritario fue

dibujar trazas de los algoritmos para conocer el comportamiento correcto que debían reconocer en SRec tras programar tal algoritmo.

- La proporción de gente que tuvo algún percance o duda durante el uso de la aplicación. Once personas (22.45%) en algún momento se quedaron temporalmente bloqueados por tener algún tipo de duda sobre la interfaz o sobre los procesos necesarios que se deben realizar con SRec para desarrollar las tareas hasta que hablaron con el profesorado. Al margen de estas once personas, otras tres tuvieron problemas con el procesado de clases Java para poder ejecutar y visualizar alguno de sus algoritmos.
- La proporción de gente que editó sus algoritmos con un editor externo (editor de texto plano, IDE…). Se pudo constatar que al menos dos personas (4,08%) usaron un editor de texto plano para editar las clases en lugar del editor integrado de SRec con coloreado de sintaxis, y otras cinco emplearon un entorno de desarrollo integrado (IDE) para realizar tales tareas, que suponen un 10,2% del total de alumnos.
- Se observó que al menos cuatro personas (8.16%) estuvieron configurando a fondo los aspectos gráficos de SRec así como la cantidad de información que podían visualizar por pantalla.
- Por último, se registraron tres incidencias en el uso de SRec. En una ocasión SRec no permitía visualizar la ejecución del código ejecutado, la cual bloqueaba el programa; en otra ocasión la visualización llegaba a generarse pero SRec impedía el manejo de la misma; y en tercer lugar, la exportación de imágenes animadas en formato GIF provocó un desbordamiento de pila por limitaciones en el uso de la memoria que hace la Máquina Virtual de Java.

7.3 Otros Tipos de Pruebas de Usabilidad

Existen tres tipos de pruebas de usabilidad, centradas en la eficiencia, que se pueden realizar:

- De exactitud: se mide el número de errores y el porcentaje de ocasiones en que los usuarios pudieron recuperarse de tales errores.

- De tiempo: se mide la cantidad de tiempo requerida para la realización de una tarea.
- De recuerdo: se mide cuántas partes y en qué grado se recuerda la aplicación pasado un tiempo desde su último uso.

Pese a que no se han realizado este tipo de pruebas, se conserva información sobre los alumnos que podría servir para realizar estas pruebas sin necesidad de poner en marcha un nuevo experimento.

Para las pruebas de exactitud y tiempo se tienen archivos de registro de actividades y eventos generados por SRec que permiten ver, con marcas de tiempo, qué hizo cada usuario durante sus sesiones en SRec, de tal forma que se podrá estudiar qué y cuántos errores se encontraron, cuántas veces tuvo que realizar cada proceso (carga de clases, compilación, creación de una visualización, etc.) y cuánto tiempo se tardó en dar por finalizada una tarea. De estos archivos, de los que se ha presentado un breve estudio inicial en el apartado "7.1 Utilización de SRec por Parte de los Alumnos", podrán inferirse datos tales como el número medio de visualizaciones creadas, el número medio de pasos dados en las visualizaciones, etc. Todos estos datos pueden suministrar datos útiles no sólo para mejorar la usabilidad, sino para conocer mejor la psicología de los alumnos y adecuar la herramienta aún más a los alumnos tomando como base los resultados de estos análisis.

Para las pruebas de recuerdo se pueden emplear los datos que se tienen sobre alumnos repetidores, viendo si su experiencia con SRec es mejor el segundo año que el primero o si el segundo año muestran un comportamiento diferente al de alumnos de primer año con los que ha compartido las mismas clases de formación en la asignatura donde se usa SRec.

7.4 Conclusiones del Capítulo

En este capítulo se ha dado una visión sobre diferentes aspectos relacionados con la usabilidad evaluados de SRec, como son la utilización de la aplicación con el fin de estudiar la eficiencia y el comportamiento de sus usuarios, que permite aproximarse a la adecuación de la funcionalidad proporcionada.

Sobre la utilización hecha de la aplicación, medida con los ficheros de registro creados por SRec, se midieron aspectos como el número de sesiones realizadas (número de veces que se abrió y cerró la aplicación), el número de clases procesadas, el número de lanzamiento de métodos, el tamaño de los vectores usados en los algoritmos que hacían uso de ellos, y, sobre todo, los errores producidos (tomando en consideración las causas y las posibles consecuencias).

Se observó que el número de errores al realizar ciertos procesos era más bajo del esperado, que los errores descendían muy notablemente a lo largo de las sesiones que los alumnos iban abriendo en SRec hasta alcanzar mínimos a partir de la sexta sesión de trabajo del alumno, un punto temprano si se tiene en cuenta que SRec es una aplicación cuya funcionalidad y forma de proceder es muy diferente a las de aplicaciones. Además, se pudo constatar que las proporciones en que se daban los distintos tipos de errores variaban en función de la tarea que tuvieran que realizar. Estos resultados permitirán afinar el proceso de depuración de la aplicación para minimizar el número de errores que los usuarios se encuentren mientras la utilicen.

En general, los resultados pueden ser considerados como muy satisfactorios al constatar que la aplicación permite realizar el trabajo de una manera fluida y sencilla, salvo en casos muy específicos que deberán ser objeto de revisión.

Además, se apreció con sorpresa el tamaño de los vectores usados para ejecutar los algoritmos, vectores pequeños y entre los que existen un amplio número que no tienen una longitud múltiplo de 2 como podría ser de esperar, pues tales vectores tienen siempre un proceso muy sencillo de división, ofreciendo como resultado subvectores de la misma longitud entre sí.

También se ha explicado el resultado de las observaciones sobre los usuarios de SRec en la quinta evaluación de usabilidad. En general, se observó que hubo un elevado número de gente que usó papel. Esto podría entenderse como la necesidad de SRec de incorporar alguna funcionalidad de predicción de valores que permita “escribir” en pantalla los valores para, a continuación, realizar la ejecución y comprobar la corrección, bien de la previsión, bien del algoritmo que se está ejecutando.

Además, se anotó el número, demasiado alto, de personas que necesitaron la asistencia del profesor para solucionar dudas acerca de la aplicación. Ello permite afirmar que inicialmente algunos usuarios se sienten algo perdidos, por lo que la

posibilidad de utilizar un asistente, al menos para los primeros usos, podría ser una posibilidad interesante para los usuarios.

También se observó la pequeña proporción de gente que usó un editor de código externo al de SRec (ya fuese un IDE o un editor de texto plano), cuya motivación para utilizarlo no se esclareció, así como el número de personas que decidieron cambiar la configuración de formato que aparece por defecto.

En definitiva, los usuarios cometieron un bajo número de errores durante el manejo de la aplicación (permite afirmar que el diseño centrado en el usuario ha logrado al menos parte de su objetivo), una parte de ellos empleó papel para poder comprobar que los resultados de SRec son los correctos (puede ser interesante introducir labores de predicción), y para algunos de ellos el editor de código es inadecuado (y susceptible por tanto de sufrir modificaciones en el futuro).

Capítulo 8. Uso Docente, Esfuerzo del Usuario y Difusión

En este capítulo se incide en diferentes aspectos de la aplicación. El primero de ellos es el uso docente, repasando el uso que se puede dar a la aplicación en distintas materias, los tipos de actividades y la adaptación de SRec a las preferencias y necesidades del profesor. Este uso siempre ha estado en el centro del trabajo, por lo que la orientación del trabajo siempre ha estado encaminada a reducir todo cuanto fuera posible el esfuerzo de creación de las animaciones.

Para facilitar la puesta en marcha de SRec en las aulas se aporta documentación en forma de manual de usuario y ayuda interactiva que son comentados en el apartado 2. Ahí se explica la creación y los contenidos de estos documentos. Gracias al manual se puede conocer exactamente cómo trabajar con SRec, qué funcionalidades están disponibles en cada momento, qué limitaciones pueden encontrarse, o cuál es la manera de proceder para obtener de SRec el mayor rendimiento.

El tercer y último apartado ofrece una perspectiva sobre las labores de difusión de SRec llevadas a cabo para intentar lograr su implantación más allá de las instalaciones y titulaciones de la Universidad Rey Juan Carlos. Para ello, se ha dotado a la aplicación de la posibilidad de la internacionalización, siendo multilingüe, de una página Web oficial, donde se puede descargar y leer documentación sobre ella, y diversas demostraciones en Congresos y Jornadas mediante pósters y presentaciones como recurso docente software. Además, se mencionará el plug-in de SRec para BlueJ implementado, que permite integrar en este IDE las funcionalidades de SRec a través de una ventana “pop-up” que aparece cuando el usuario de BlueJ lanza la ejecución de un algoritmo, permitiéndole hacer un uso de las visualizaciones similar al de la versión “stand-alone” de SRec.

8.1 Uso Docente

En este apartado se recogen ciertas capacidades de SRec orientadas al mundo docente.

Una de estas características es la posibilidad que ofrece SRec de cargar y guardar visualizaciones en formato XML. Esto permite al profesor realizar una librería

de visualizaciones que siempre están disponibles para ser utilizadas con la aplicación. Cada una de estas visualizaciones puede tener una configuración de formato diferente, y estar ubicada en un punto estratégico de la animación, de tal forma que estas visualizaciones permiten, en pocos segundos, ofrecer ejemplos interesantes durante las clases.

Además del guardado y cargado de animaciones se permite la exportación en formatos gráficos estándar, de tal forma que los materiales resultantes pueden ser empleados en transparencias, informes de prácticas, apuntes, páginas Web… y cualquier otro tipo de material didáctico, generado tanto por profesores como por los propios alumnos.

Además, las opciones que filtran qué información se muestra y cuál no permiten ofrecer diferentes uso de la aplicación incluso empleando el mismo algoritmo y la misma ejecución de éste, de tal forma que profesores y alumnos pueden centrarse en diferentes tareas según sus intereses y necesidades. La búsqueda y selección de llamadas recursivas refuerzan esta funcionalidad.

Así, SRec puede ser empleado mientras se estudian diversos aspectos de la algoritmia, como la complejidad en tiempo, la complejidad en memoria o la redundancia que incorporan los algoritmos estudiados.

Las múltiples opciones de configuración de formato que brinda SRec permiten que éste pueda adaptarse al tipo de pantalla sobre la que está siendo empleado (proyector en el aula, monitor personal…). La capacidad de mostrarse en múltiples idiomas permite cubrir la demanda que pudiera surgir desde países que no sean hispanohablantes.

8.2 Documentación

SRec se proporciona junto con cierta documentación que facilita al usuario el conocimiento y utilización de la aplicación. Esta documentación está formada por el Manual de usuario y la ayuda interactiva integrada en la ventana de SRec. Ambos son brevemente descritos a continuación.

8.2.1 Manual de Usuario

El Manual de Usuario de SRec es un documento que se proporciona en soporte electrónico que permite conocer en profundidad el funcionamiento, las propiedades, las funcionalidades y la manera de proceder de SRec. El objetivo del documento facilitar al usuario final los primeros contactos con la aplicación, permitiéndole conocer qué características están disponibles en cada momento y cómo activar cada una de las funcionalidades que ofrece la aplicación. No existen diferentes versiones del manual para profesores y alumnos puesto que las funcionalidades que pueden utilizar ambos grupos de usuarios son prácticamente idénticas. El Manual de Usuario de SRec consta de siete apartados que son resumidos a continuación:

- Apartado 1: Visión general de la aplicación.
- Apartado 2: Instalación de SRec.
- Apartado 3: Visualizaciones.
- Apartado 4: Configuración de visualizaciones.
- Apartado 5: Lectura y edición de código.
- Apartado 6: Otras opciones.
- Apartado 7: Disponibilidad de SRec.

Además del Manual de Usuario de SRec, se han desarrollado dos guías rápidas, una sobre el proceso de instalación y otra sobre el procesamiento de clases y generación de visualizaciones. El objetivo de éstas es simplificar y esquematizar el aprendizaje de los principales procesos (origen de diversos problemas) para agilizar el arranque de actividad productiva con SRec por parte de los alumnos. A diferencia del manual de usuario, las guías rápidas sólo han sido distribuidas entre alumnos en el momento de realizar la quinta evaluación de usabilidad.

8.2.2 Ayuda Interactiva de SRec

La Ayuda Interactiva de SRec está disponible a través del menú Ayuda de la aplicación y permite acceder a una información muy similar a la que ofrece el Manual de Usuario. El propósito de la ayuda no es tanto ayudar a dar los primeros pasos sino servir de apoyo ante una duda o problema puntual del usuario mientras emplea la aplicación aun en el caso de que lleve usándola un tiempo prolongado.

Este servicio es mucho más cómodo, rápido y directo que el manual, pues puede ser activado desde la propia ventana de SRec y permite una navegación mediante hipervínculos, al estilo de una página Web, logrando una navegación ágil y directa hacia los contenidos de interés.

8.3 Difusión de SRec

A lo largo de estos años se han realizado diversas labores de difusión de la aplicación para darla a conocer en diversos foros y jornadas dedicadas a la educación de la informática y facilitar su utilización más allá del entorno propio de la Escuela Técnica Superior de Ingeniería Informática de la Universidad Rey Juan Carlos.

Una de ellas ha consistido en permitir que SRec esté disponible en más de un idioma, como se verá en el primer subapartado. Otra de las tareas desempeñadas para darlo a conocer fue abrir una página Web que sirva de escaparate de la aplicación.

También ha sido llevado SRec a encuentros con profesionales de la enseñanza como Congresos y Jornadas para divulgar su existencia y cualidades así como recibir cierta información de retroalimentación en forma de sugerencias, críticas e ideas.

Como tarea de difusión de SRec también encaja el desarrollo del plug-in para BlueJ, que será presentado en el apartado “8.3.4 Plug-in de SRec para BlueJ“, pues permite dar a conocer SRec entre los usuarios de BlueJ.

8.3.1 Internacionalización de SRec

Si al viajar al extranjero lo primero que una persona percibe es un idioma diferente, a una aplicación software le ocurre algo parecido; lo que encontrará será, en la mayoría de ocasiones, potenciales usuarios que hablan un idioma distinto.

Es por ello que resulta fundamental dotar a la aplicación de la capacidad de adaptarse al idioma que manejan los usuarios. SRec cuenta con un flexible sistema que le permite ofrecerse en un número ilimitado de idiomas, fácilmente incorporables mediante la edición de un simple documento XML, tal y como ya se explicó en el apartado “4.3.5 Documento XML para los Textos en Múltiples Idiomas”.

SRec se proporciona actualmente en dos idiomas, español e inglés, cubriendo así la inmensa mayoría de la población potencial a la que va dirigido, sobre todo gracias al inglés. No obstante, se permanece abierto a colaboraciones que permitan integrar otros

idiomas en SRec con el fin de hacer más cercana la aplicación a otros países y más cómoda su utilización a sus habitantes.

En cualquier momento el usuario puede cambiar el idioma de la aplicación. Todos los elementos pasarán a estar legibles en el idioma elegido tras unos segundos en que SRec cambia el idioma de tales elementos.

8.3.2 Página Web

SRec aprovecha la mayor ventana de comunicación que existe actualmente: Internet. El objetivo no es otro que darse a conocer para promover su uso en la comunidad docente, ya sea en el ámbito universitario, en cursos académicos y/o de otros niveles o para autodidactas independientes. Este servicio emplea actualmente los recursos proporcionados por el grupo de investigación LITE, al que pertenecen los autores de esta aplicación. La dirección URL de acceso a la misma es: http://www.lite.etsii.urjc.es/srec. En la Ilustración 52 aparece una captura de su página de presentación.

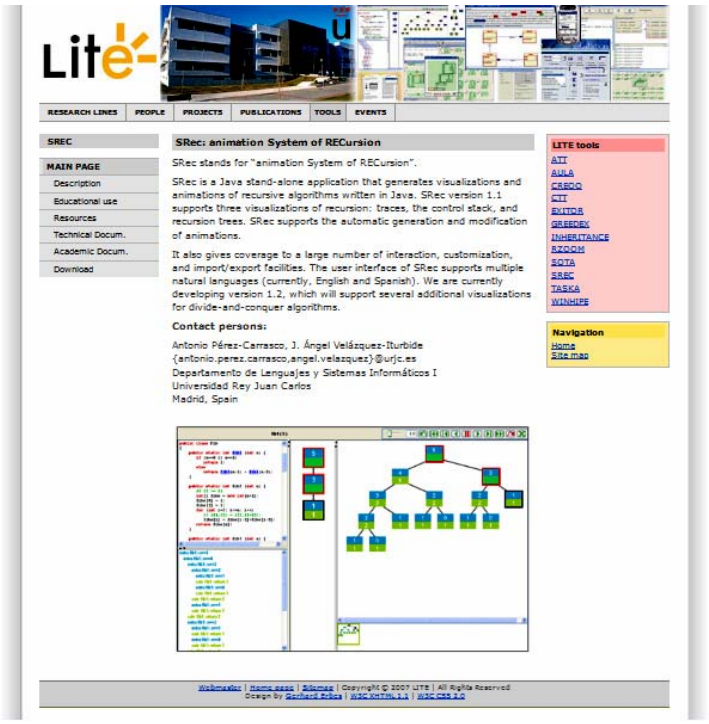


Ilustración 52. Página de portada de la Web de SRec

Para facilitar su difusión a un mayor número de personas, la página Web se presenta en inglés, idioma empleado por defecto en los círculos de investigación a nivel internacional.

La página Web ofrece información básica sobre la herramienta, enlazando publicaciones técnicas sobre SRec y haciendo referencia a determinadas publicaciones de investigación que pueden ayudar a comprender mejor la utilidad y características de la aplicación. Los apartados en que se divide la página Web son los siguientes:

- Página inicial: se ofrece una descripción básica, aportando además medios de contacto con los autores y una captura de la ventana de la aplicación, con el fin de ayudar a ofrecer una primera imagen general de SRec.
- Descripción: se detallan las características de SRec, haciendo hincapié en las posibilidades que otorga la barra de animación, las posibilidades de los paneles de visualización, el procedimiento para generar visualizaciones, funciones de importación y exportación, funciones de personalización de la configuración, características de muestreo de información y otras propiedades.
- Uso educativo: se detallan las funciones educativas que se pueden ejercer (comprensión del comportamiento recursivo de los algoritmos, análisis de eficiencia de los algoritmos, análisis de la redundancia de cálculos en algoritmos con recursividad múltiple), los tipos de asignaturas en las que se puede usar (programación, algoritmia) y los modos en que puede ser empleado (durante clases teóricas en el aula, autoestudio en casa, prácticas en el laboratorio o con la intención de creación de documentación).
- Recursos: se ofrecen algunos algoritmos de prueba para ser ejecutados y visualizados con SRec, como los números de Fibonacci, la función de Ackermann, o números combinatorios.
- Documentación técnica: se informa de los requisitos de SRec para poder funcionar correctamente y se enlazan documentos técnicos relacionados con la aplicación.
- Documentación académica y de investigación: aparecen las referencias a diversas contribuciones publicadas en las Actas de Congresos y en revistas, así como a las demostraciones de software realizadas y otros logros.
- Página de descarga: incluye un breve formulario que los interesados deben rellenar para poder descargar la aplicación donde se les pide información sobre su nombre, centro de estudios, una dirección de correo electrónico, un

perfil (estudiante / profesor / otro) y el propósito de su descarga (usar en un curso / aprendizaje independiente / curiosidad).

Desde el 14 de enero de 2010 hasta el 14 de julio de 2011 se obtuvieron los datos estadísticos sobre las 22 descargas registradas de SRec que aparecen en la Tabla 38, la Tabla 39, la Tabla 40 y la Tabla 41 basados en la información suministrada en el formulario de la página:

Uso de la aplicación	**Para un curso**	**Aprendizaje indep.**	**Sólo curiosidad**
	63,6% (14)	27,3% (6)	9,1% (2)

Tabla 38. Uso que le darán a SRec las personas que lo descargaron.

Perfil del visitante	**Profesor**	**Estudiante**	**Indeterminado**
	36,3% (8)	54,5% (12)	9,2% (2)

Tabla 39. Perfil de las personas que descargaron SRec.

Origen del visitante	**URJC / España**	**Europa**	**América**	**Asia**	**Sin identificar**
	27,27% (6) 13,63% (3)	0,0% (0)	36,36% (8)	4,55% (1)	18,18% (4)

Tabla 40. Origen de las personas que descargaron SRec.

Meses con más descargas	**Abril**	**Junio**	**Febrero**	**Mar, Oct, Dic**	**Ene, May, Sep, Nov**
	22,73% (5)	18,18% (4)	13,64% (3)	9,09% (2)	4,55% (1)

Tabla 41. Meses con un mayor número de descargas de SRec.

8.3.3 Demostraciones de SRec

SRec ha sido mostrado en dos encuentros científicos y de divulgación como son ITICSE (en su edición de 2009) y JENUI (también en su edición de 2009).

El primero de ellos, Annual Conference on Innovation and Technology in Computer Science Education, tiene carácter internacional y está catalogado en la lista CORE, categoría A. En él SRec y sus por entonces novedosas características de visualización de la técnica “divide y vencerás” fueron presentados a través de un póster, si bien en la edición anterior de ITICSE SRec logró estar presente mediante un artículo largo para dar una vista general de su funcionalidad.

Por otra parte, el segundo de ellos, las Jornadas de Enseñanza de la Informática, se produce a nivel nacional y pretende ser un punto de encuentro entre profesores de informática, generalmente del ambiente universitario. En estas Jornadas se presentó una versión general de SRec como recurso docente para continuar las labores de difusión

llevadas a cabo hasta entonces y recibir sugerencias y críticas de parte de colegas en un ambiente cercano.

Con estas dos demostraciones se consiguió cubrir distintos tipos de público que potencialmente podría estar interesado en integrar SRec en sus clases de algoritmia. Los objetivos fundamentales, como ya se ha explicado, fueron divulgar las propiedades de la herramienta, recopilar opiniones, críticas y sugerencias desde un punto de vista de expertos, así como establecer contactos con investigadores que se encuentren desarrollando trabajos similares con los que poder intercambiar experiencias, conocimientos e inquietudes.

8.3.4 Plug-in de SRec para BlueJ

A día de hoy los entornos integrados de desarrollo software (conocidos como IDE) son fundamentales para facilitar tanto las tareas de programación como las de enseñanza de la programación. Algunos de ellos son sencillos, sobre todo si están orientados a la docencia, y otros toman una forma más compleja, como los que cuentan con un enfoque más profesional.

En cualquiera de los casos, es común encontrar entornos que admiten la integración de librerías o aplicaciones externas con el fin de aumentar la funcionalidad y las posibilidades del entorno. Con estas librerías o aplicaciones integradas se consigue obtener toda la funcionalidad necesaria empleando una única herramienta, conociendo un único sistema y sin necesidad de buscar o elaborar una adecuada sincronización entre distintas aplicaciones que logren obtener el fin deseado.

A lo largo de este apartado se presenta el plug-in desarrollado y su motivación, las características del mismo, y de manera muy breve BlueJ.

8.3.4.1 Motivación

En la actualidad, ya existen muchísimos plug-ins para entornos integrados de desarrollo que ofrecen diversas ventajas frente a las versiones que funcionan de manera independiente.

Eclipse, por ejemplo, es un entorno integrado de desarrollo de carácter profesional que es empleado también en multitud de escuelas universitarias. Para Eclipse existen plug-ins de carácter docente [43][53], como el de AnimalScript [77],

que permite editar código de script para Animal [76], una versátil herramienta para la creación, modificación y presentación de visualizaciones y animaciones de algoritmos.

La realización del plug-in es ampliamente beneficioso para todas las partes implicadas como ya se ha indicado: las aplicaciones junto a sus desarrolladores y los usuarios de las mismas. Es ésta la principal motivación, que queda desarrollada a continuación.

La existencia de este plug-in beneficia en primer lugar a SRec, puesto que le permite ganar notoriedad al aumentar su difusión al presentarse ante los usuarios de BlueJ. Así, la comunidad de usuarios de SRec podrá crecer al quedar accesible entre los recursos, librerías y plug-ins de que disponen los usuarios de BlueJ [36].

Por otra parte, SRec ayuda a enriquecer BlueJ, que cuenta ya con un amplio catálogo de ampliaciones externas que añaden su funcionalidad a las que ya tiene este IDE. Este repertorio de plug-ins permite darle un valor añadido a BlueJ, pues éste queda personalizado por parte del usuario con las librerías que le interesa emplear para ajustarlo a sus necesidades.

Los usuarios no se enfrentan a la obligación de cambiar de herramienta para poder estudiar la ejecución de los algoritmos con las vistas de SRec. Hasta ahora, si querían emplear SRec para visualizar los algoritmos recursivos, tenían que desarrollar sus programas en BlueJ y posteriormente cargar la clase correspondiente en SRec para poder visualizar el algoritmo deseado. Otra opción era escribir el código directamente en SRec, que cuenta con un cuidado editor pero que, obviamente, no ofrece tantas posibilidades como el de BlueJ, al no ser ésta su función principal. Esto requiere de un esfuerzo adicional por parte del alumno, pues éste tiene que aprender a manejarse con SRec si es que hasta ahora no lo conocía.

Por tanto, los usuarios ya no tienen que conocer cómo desarrollar código en el visualizador de recursividad SRec ni molestarse en transportar los programas escritos desde una aplicación a otra, con la consiguiente ganancia de tiempo.

8.3.4.2 BlueJ y plug-ins para un IDE

BlueJ [37] es una aplicación orientada a dar los primeros pasos en el aprendizaje de la programación y de la orientación a objetos haciendo uso del lenguaje Java. Cuando el usuario inicia la aplicación y abre un proyecto de trabajo se habilita en la vista principal el diagrama UML que representa las relaciones entre las distintas clases

que componen dicho proyecto. En ese preciso instante el usuario puede comenzar a interactuar directamente a través de la interfaz con las clases para crear objetos o ejecutar alguno de sus métodos.

BlueJ ofrece a los desarrolladores de software una API que facilita la captura de eventos como la compilación de una clase, la edición de código, la ejecución de un método, etc. de tal forma que sincronizar lo que ocurre en BlueJ con una segunda aplicación añadida en forma de plug-in resulta muy sencillo. No extraña por tanto el amplio número de plug-ins disponibles para BlueJ que se encuentran enlazados desde la página Web de tal aplicación [36], lo cual ha permitido a BlueJ ganar en funcionalidad, aplicabilidad e implantación, llegando a ser usado regularmente en más de un millar de centros por todo el mundo.

Entre ellos destacamos por sus semejanzas con SRec el plug-in de Jeliot [47]. Ésta es una aplicación “stand-alone” que permite introducirse en el mundo de la programación haciendo uso del lenguaje Java [46]. Ofrece, mediante la metáfora de un teatro, la posibilidad de reproducir de manera animada la ejecución de algoritmos de diversa complejidad.

8.3.4.3 Instalación y uso del plug-in

La instalación del plug-in realizado es muy sencilla. Basta descomprimir el archivo ZIP en el que se obtiene en la carpeta adecuada de la instalación de BlueJ para que ésta pueda encontrar los componentes y poder hacer uso de ellos durante su ejecución.

El archivo ZIP contiene el archivo ejecutable de SRec preparado específicamente para funcionar como plug-in de BlueJ (no soporta la utilización como aplicación “stand alone”). Una vez que el contenido del archivo ZIP ha sido adecuadamente ubicado en la carpeta de extensiones de BlueJ, éste lo añadirá a su funcionalidad automáticamente. No obstante, es posible que se desee desactivar y activar el plug-in en determinados momentos de utilización de BlueJ. Para ello basta dirigirse al menú “Tools” de BlueJ para seleccionar la opción de activación y desactivación de SRec.

Una vez que el plug-in está instalado y activado, su uso es muy sencillo. Cada vez que BlueJ ejecute un método de una determinada clase, SRec lo ejecutará también para iniciar su ulterior visualización. Si la ventana de SRec no está visible, ésta

aparecerá automáticamente. Si la ventana de SRec ya contiene una visualización creada previamente, ésta será descartada y se pasará a mostrar la nueva visualización.

La ventana de SRec podrá ser cerrada en cualquier momento por el usuario si éste no desea seguir visualizando un algoritmo (si bien volverá a abrirse automáticamente en caso de que se lance un nuevo método). Cuando no se desee hacer uso por más tiempo de BlueJ (con o sin el plug-in de SRec), bastará cerrar la ventana del primero, pues la ventana de SRec se cerrará instantáneamente tras realizar esa acción.

Las posibilidades de interacción con las visualizaciones que muestra el plug-in de SRec para BlueJ son idénticas a las de la versión "stand alone", de tal forma que se puede desarrollar el mismo trabajo con ambas versiones. No obstante, esta primera versión del plug-in no activa vistas adicionales para la técnica de "divide y vencerás" como sí es capaz de hacer la versión "stand alone". Se muestra en la Ilustración 53 el diagrama de estados de las dos aplicaciones.

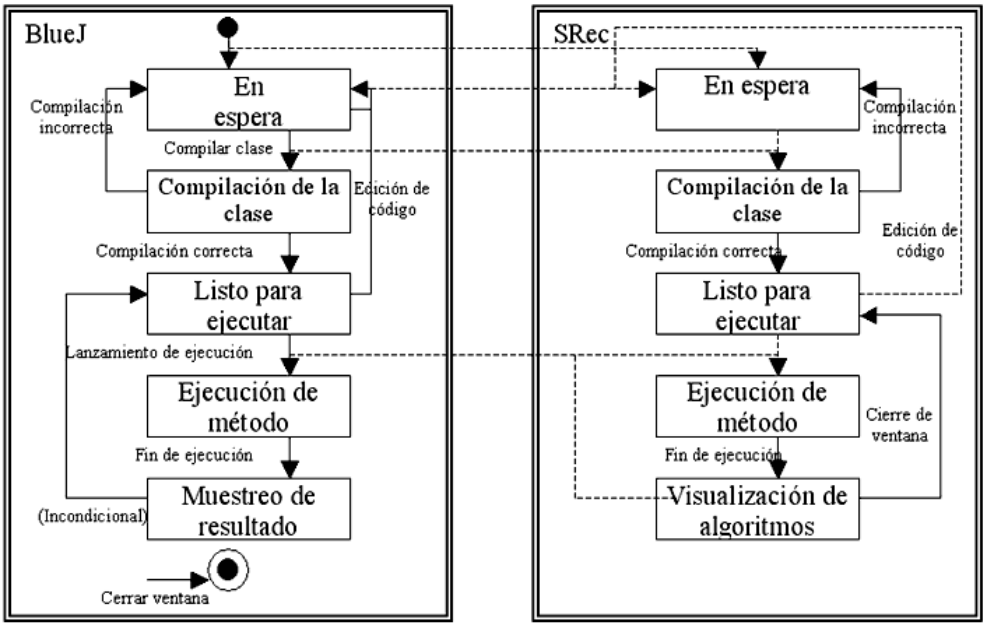


Ilustración 53. Diagrama de estados de BlueJ y el plug-in de SRec

No obstante, algunas opciones de funcionamiento están deshabilitadas, como las de carga y edición de clases, pues el comportamiento del plug-in está supeditado a las clases que maneja BlueJ mediante su propia ventana, siendo éste el que indica al plug-in qué clases cargar y qué métodos ejecutar así como con qué parámetros. Es por ello que el usuario no verá en ningún momento el código de la clase a través de la ventana de SRec, pues podrá verlo y editarlo a través de BlueJ, como refleja la Ilustración 54.

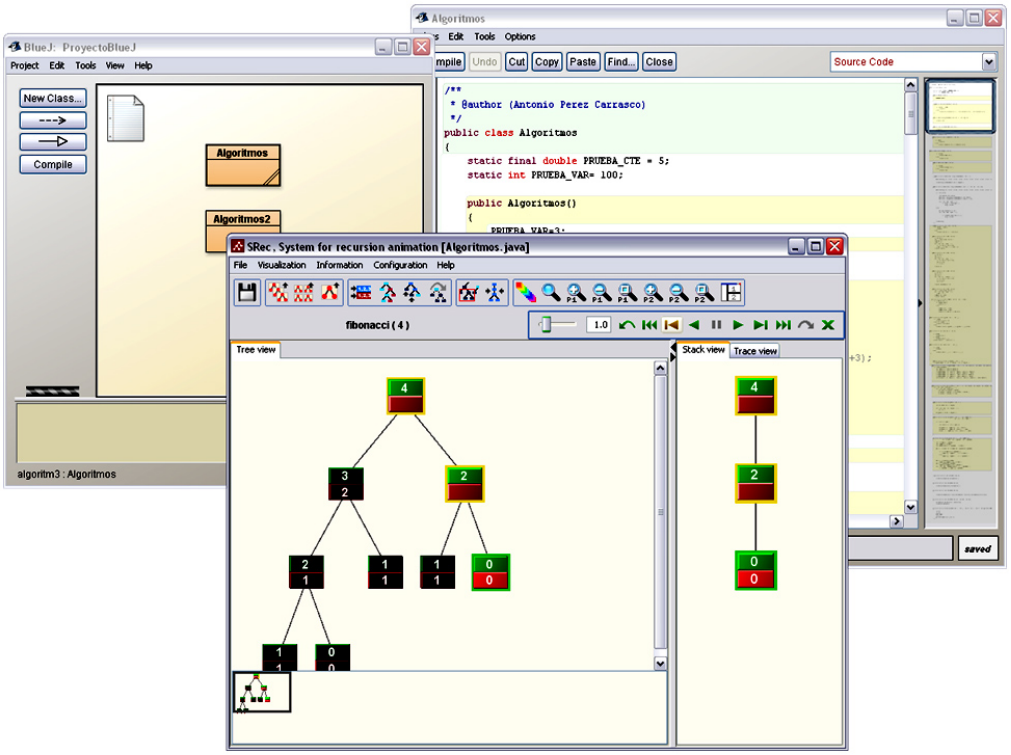


Ilustración 54. BlueJ y SRec trabajando de manera conjunta

8.4 Conclusiones del Capítulo

En este capítulo se ha hecho un repaso de aspectos diversos. En primer lugar se han repasado algunas cualidades de la aplicación orientadas a su uso docente. Se considera que se han cumplido los objetivos de otorgar una serie de facilidades docentes de tal forma que se permiten conseguir objetivos docentes más cómoda y fácilmente gracias a la aplicación por medio de sus visualizaciones, la capacidad de almacenamiento de las mismas y su flexibilidad de configuración, adaptándose a diferentes formas de trabajo, entornos y tipos de usuario.

También se ha revisado la documentación que se le proporciona al usuario para ayudarle a conocer mejor la aplicación. Para este fin se le proporciona un completo manual de usuario y una ayuda interactiva, cada uno de ellos tiene un modo de uso distinto. El suministro de este material junto con las guías rápidas logra aliviar ciertos problemas iniciales con la aplicación, consiguiendo así una mejor aceptación y un uso mucho más productivo.

Además, se ha explorado la creación de un plug-in de SRec para el IDE BlueJ, explicando la motivación y las ventajas que supone este plug-in para los usuarios así

como para las propias aplicaciones. El plug-in aún no está publicado para su utilización porque están siendo revisados algunos aspectos meramente técnicos, pero se cree firmemente que supondrá una importante ayuda a la hora de aumentar la implantación de SRec en las aulas.

Por último, se ha expuesto el conjunto de labores realizadas para cumplir con el objetivo de difusión de la aplicación: internacionalización, creación de página Web y demostraciones de SRec en encuentros científicos de distinto rango. Se considera que tales labores han comenzado a ofrecer frutos, pues la página Web ha detectado descargas desde varios continentes, constatando así la utilidad de la propia página Web y la conveniencia de la internacionalización de la aplicación, suponiendo a su vez que las demostraciones y presentaciones de SRec en diferentes foros científicos ha sido la vía por la que se conoció la existencia de la aplicación por parte de quienes posteriormente decidieron visitar la página de la aplicación y efectuar su descarga.

En cualquier caso, el éxito de descargas (y por tanto de expansión de la aplicación) ha sido moderado pese a las labores de difusión realizadas. Cabe preguntarse por tanto qué ha motivado que no se consiga un mayor nivel de éxito en cuanto a notoriedad internacional y si seguir realizando las labores de difusión igual que hasta ahora puede tener o no resultados prometedores de cara al futuro.

De esta forma, culmina un trabajo en el que se ha pretendido en todo momento ofrecer una aplicación que solucione ciertas carencias, de una manera sencilla, ágil y que tenga como resultado un mejor aprovechamiento de las clases presenciales y del tiempo de estudio del alumno, siguiendo una serie de criterios explicados y marcados a menudo como objetivos para terminar dándole difusión y apostar por la implantación de la aplicación en múltiples centros de estudio.

Capítulo 9. Conclusiones

A lo largo de esta memoria de Tesis Doctoral ha quedado desglosado el trabajo realizado por el doctorando Antonio Pérez Carrasco. Con el material expuesto en esta memoria se puede concluir que se presenta un trabajo informático que consta de las fases habituales: localización de un problema para resolver, estudio del arte relacionado, desarrollo de una aplicación software que lo resuelve, evaluación de la misma y difusión de la aplicación para su implantación.

En el Capítulo 1 se ha presentado una introducción al trabajo realizado, aportando una motivación, una hipótesis de inicio, y unos objetivos que se deseaban cubrir, mencionando además las aportaciones principales realizadas. Se ha expuesto, a través del Capítulo 2, un completo estado del arte, repasando el momento en que se encuentra la visualización de la información y las disciplinas relacionadas con el trabajo realizado: la visualización del software, la visualización de programas y la visualización específica de la recursividad. También se ha explorado el repertorio de modelos conceptuales existentes sobre la recursividad, su utilización en la bibliografía y los más habituales modelos mentales sobre ese concepto.

Tanto el Capítulo 3 como el Capítulo 4 se centraron en describir la aplicación SRec desde diversos puntos de vista. El primero de esos capítulos dio una perspectiva global, expuso las características de la interfaz, presentó las representaciones gráficas que SRec genera, su capacidad de animación y sus opciones de interacción. El segundo está centrado en características de implementación, revisando la arquitectura interna para la generación de las vistas, el flujo de datos que se maneja mientras se procesan clases de código Java para su posterior visualización, o la utilización que se hace del lenguaje de marcado XML.

El Capítulo 5 desarrolla el análisis de la eficacia de SRec respecto a la visualización. Por un lado se compara SRec con otras aplicaciones orientadas a la visualización de la recursividad, hallando una amplia superioridad de SRec frente a las demás en términos de posibilidades de interacción y uso docente, esfuerzo de generación de animaciones, número de modelos conceptuales representados y catálogo de controles de animación. De esta forma se logran identificar las mejoras que introduce SRec respecto a los sistemas existentes y la aportación técnica con fines docentes que se

realiza. Por otra parte, también se evalúa a SRec desde el punto de vista de la capacidad ilustrativa de algoritmos de la técnica “divide y vencerás”. Para ello se estableció una catalogación de problemas donde se pudo apreciar para qué grupos de problemas SRec ofrece una respuesta ampliamente satisfactoria y para qué otros grupos se deben implementar ciertas mejoras, situadas en un contexto de trabajos futuros.

Por su parte, el Capítulo 6 y el Capítulo 7 reflejan las distintas evaluaciones de usabilidad que se han realizado sobre SRec. Por un lado se recogen todos los experimentos realizados con alumnos sobre la evaluación de la usabilidad que han tenido lugar durante cinco cursos académicos, recogiendo los datos mediante cuestionarios. De estos se pudieron extraer valoraciones numéricas, sugerencias y críticas, que ayudaron en el desarrollo ulterior de SRec una vez se reflexionó sobre los datos recopilados. El segundo de estos capítulos expone un primer estudio sobre el uso de la aplicación (número de clases procesadas, número de errores generados, tamaño de los ejemplos probados…) y también un amplio resumen de las observaciones realizadas sobre los usuarios observando su actitud y acciones complementarias como uso de papel.

Por último, el Capítulo 8 refleja el uso docente que se puede realizar de SRec. Se presenta también la documentación orientada al usuario (manual y ayuda interactiva) que permite agilizar la implantación de la aplicación. A continuación se abordan las diversas labores de difusión llevadas a cabo, como la capacidad de internacionalización de SRec, la página Web con la que cuenta, las demostraciones realizadas en distintos seminarios y jornadas y el desarrollo de un plug-in desarrollado para el IDE BlueJ.

Las características de la implementación permiten a SRec ser ampliado fácilmente sin esfuerzo para el desarrollador, lo que puede facilitar que el proceso de desarrollo sea realizado por terceros, que podrán asimilar el código que lo implementa fácilmente. Además, el empleo del lenguaje XML puede facilitar una posterior extensión que permita a SRec comunicarse con otras aplicaciones y emplear otros lenguajes de representación gráfica basados en XML como podría ser SVG para su visualización con navegadores estándar.

Por otra parte, los experimentos de evaluación de usabilidad han permitido realizar un desarrollo de una herramienta afín a los gustos y necesidades de los usuarios, no sin encontrarse en ocasiones ciertos desajustes que han provocado que las

valoraciones numéricas dadas por los participantes en las evaluaciones no fueran todo lo altas que pudiera desearse. No obstante, y a pesar de las críticas y sugerencias recibidas, los alumnos expresaban espontáneamente lo mucho que les había ayudado SRec en sus tareas a la hora de comprender y depurar más fácil y rápidamente los algoritmos, éste no dejará de ser nunca el objetivo fundamental de SRec, que se considera ampliamente conseguido.

Las diferentes evaluaciones de SRec servirán para mejorar la aplicación de cara al futuro. La evaluación del uso de SRec que tiene lugar por parte de los alumnos y cómo se comportan mientras la utilizan permitirá investigar cuáles son las partes más problemáticas o con más carencias, mejorarlas, y evitar que los usuarios choquen con la aplicación, suavizando a su vez la curva de aprendizaje gracias a las mejoras o ampliaciones de las funcionalidades.

Además, la evaluación de la adecuación para la técnica “divide y vencerás” ha permitido identificar las clases de problemas para los que SRec es menos capaz de ofrecer visualizaciones de alto valor ilustrativo, por lo que ahora se podrá investigar sobre cómo mejorar estas carencias mediante nuevas vistas y/o mejorando las ya existentes.

9.1 Satisfacción de la Hipótesis

Tras la realización de este trabajo se considera que se ha satisfecho la hipótesis de partida, que planteaba la realización de una aplicación con fines docentes, propiedades de interacción y de carácter automático. Estos tres aspectos se desarrollan a continuación.

Efectivamente, la aplicación construida tiene fines docentes pues se enmarca en la enseñanza de la informática, específicamente en la de la recursividad y la técnica de diseño de algoritmos “divide y vencerás”. Mediante visualizaciones los alumnos pueden comprender, analizar y depurar algoritmos recursivos y aquellos que han sido diseñados bajo la técnica “divide y vencerás”, cumpliendo así la función didáctica.

Además, la aplicación cuenta con diversas opciones de interacción y control de la información que se muestra en pantalla. De esta forma, el usuario puede ampliar información sobre los elementos que está visualizando, filtrar datos, explorar los elementos disponibles, reconfigurar la disposición de la información o su formato,

aplicar diferentes niveles de abstracción, seleccionar la vista más adecuada en cada momento, dar pasos de diferente magnitud en la navegación en el tiempo, etc. Así, se ofrece una amplia gama de posibilidades de interacción que aumentan la implicación del alumno.

Por otra parte, la aplicación cuenta con un carácter automático que consigue que la generación de las visualizaciones se realicen con muy pocos clics, justo los necesarios para elegir la clase Java que contiene el algoritmo, qué método de los contenidos en la citada clase se desea visualizar, e insertar los valores de entrada que se necesitan. La generación y gestión de los elementos que aparecen en las visualizaciones queda por tanto a cargo del programa, sin que el usuario tenga que encargarse de nada.

De esta forma, queda confirmada la hipótesis propuesta al comienzo de este trabajo, demostrando que es posible desarrollar una aplicación software con fines docentes, propiedades de interacción y de carácter automático de manera exitosa, efectiva y respetando los principios del diseño centrado en el usuario.

9.2 Aportaciones

A nivel técnico, se han realizado varias aportaciones al panorama de la visualización de programas con fines docentes.

En primer lugar se aporta el desarrollo de una aplicación software que trae consigo una serie de capacidades y funcionalidades que hasta ahora no estaban disponibles para la comunidad docente. La aplicación hace uso de un enfoque general, sin limitarse a un catálogo preestablecido de ejemplos predefinidos, permitiendo así al usuario ejecutar y visualizar cualquier programa recursivo que diseñe.

La generación de las visualizaciones es totalmente automática. Apenas bastan un par de clics para elegir qué se debe ejecutar y con qué parámetros de entrada. Con ello la aplicación genera la visualización basada en múltiples vistas, hasta un total de cinco según los casos, lo cual no es habitual en este tipo de aplicaciones.

SRec proporciona dos vistas que hasta ahora no habían sido implementadas en ninguna aplicación para la visualización de un tipo concreto de algoritmos, los diseñados bajo la técnica “divide y vencerás”. Estas dos vistas, centradas en la

estructura de datos manejada por el algoritmo suponen por tanto un importante elemento diferenciador entre SRec y las demás aplicaciones.

Por otro lado, la flexibilidad en el manejo de las animaciones es también una de las señas de identidad de SRec, que amplía de forma muy notoria el rango de pasos que se pueden realizar desde cada instante que se muestra de la visualización. Saltos, pasos simples, o avances hasta el final, todos ellos en ambas direcciones, suponen el más amplio catálogo de controles encontrado en las aplicaciones educativas existentes. Por si fuera poco, SRec cuenta con un amplio catálogo de opciones de interacción (como el marcado de nodos, la ampliación de información sobre un elemento, filtrado de datos, reconfiguración de los elementos, etc.) que son ampliamente superiores en cantidad y funcionalidad a las de las demás aplicaciones existentes.

Como ya se ha explicado ampliamente, la aplicación contiene una serie de funciones docentes que ayudan al profesor y estudiante activamente en sus labores. Estas funciones no están en un gran número de aplicaciones existentes, por lo que la capacidad de estas queda limitada frente a las posibilidades de SRec. No debe olvidarse que SRec proporciona ciertas facilidades de instalación (archivo autoejecutable, guías rápidas) que permiten mitigar las reticencias a ser utilizado en el contexto docente.

Durante este periodo de trabajo, como ya se mencionó en el Capítulo 6, se ha realizado un exhaustivo y prolongado estudio de usabilidad que ha permitido ver y analizar la evolución de la aceptación de SRec entre los alumnos a medida que se han ido incorporando y mejorando las funcionalidades. Este diseño centrado en el usuario ha permitido mejorar la aplicación conforme a los intereses de los usuarios, lo cual es muy poco habitual en sistemas de visualización.

9.3 Trabajos Futuros

Existen una serie de trabajos, algunos con una componente meramente técnica mientras que otros conservan un carácter más investigador, que serán realizados desde 2012 de una manera más profunda e independiente, pero permitiendo que los logros de uno incidan positivamente en otros. Todos ellos se listan a continuación.

Uno de los trabajos futuros que se realizarán próximamente será un estudio de la eficacia educativa de SRec. Durante este trabajo se ha empleado un enfoque técnico, centrado en el desarrollo de la aplicación siguiendo los objetivos iniciales matizados por

las principales convenciones sobre usabilidad y la realimentación obtenida en las evaluaciones de usabilidad realizadas. Sin embargo, es igualmente importante el estudio de la calidad educativa de la aplicación. Una vez alcanzado el momento en que se tiene una aplicación bien diseñada se dan las circunstancias adecuadas para llevar a cabo un estudio profundo de eficacia educativa que permita cuantificar si los alumnos aprenden más, mejor, más rápido o con mayor motivación empleando SRec.

Un trabajo que se encuentra aún en su fase inicial y que se espera que tenga un gran valor es el estudio integral de la información recopilada durante la quinta evaluación de usabilidad. Así, y como ya se ha comentado, se recopiló información sobre cada participante en cuatro formas:

- Uso de la aplicación.
- Comportamiento del alumno.
- Material académico entregado.
- Cuestionario.

Con todos estos datos podrán establecerse perfiles de usuarios, encontrarse causas y soluciones para diversos problemas en la aceptación de SRec, y cómo influyen los problemas que puedan derivarse del uso de SRec en el rendimiento académico.

Una de las facilidades educativas que puede resultar interesante es la de la predicción y simulación [38]. Por ello, se va a estudiar la introducción en SRec de un modo predictivo que, tras ejecutar el algoritmo correspondiente, en lugar de empezar a mostrar la visualización, dé al usuario la posibilidad de generarla él, revisando por parte de SRec paso a paso la corrección de los valores introducidos e intentando explicar los fallos detectados. De esta manera podría evitarse la utilización de papel registrada en la quinta evaluación de usabilidad realizada.

La arquitectura interna de SRec permite generar fácilmente un número ilimitado de visualizaciones dependiendo del tipo de algoritmo que se esté ejecutando. Por ello, puede ser muy interesante no limitar SRec a sólo la visualización de la recursividad y la técnica “divide y vencerás”, de tal forma que ya existen planes para comenzar a integrar técnicas de diseño de algoritmos como la “programación dinámica”. Esto permitiría aumentar el periodo de uso de SRec en las asignaturas y dotarlo de mayor funcionalidad

y utilidad educativa, consiguiendo a la vez rentabilizar más el aprendizaje necesario para usar la aplicación adecuadamente.

Por otra parte, se mejorarán las vistas para la técnica “divide y vencerás” para ilustrar todo tipo de algoritmos diseñados bajo esta técnica. Una de las clases de algoritmos tratadas en este estudio, la de los algoritmos de carácter geométrico, podría encontrarse con una nueva vista en SRec específica del dominio, pero con la suficiente generalidad como para ser usada con todos los problemas de la citada clase. Esto abriría la puerta a la introducción de algoritmos de mayor nivel en las aulas, como los problemas multidimensionales de la técnica “divide y vencerás”. Este trabajo ya cuenta con diversas propuestas presentadas [69] que serán maduradas e implantadas en la aplicación en el futuro.

En definitiva, será fundamental profundizar en el estudio global que maneja diferentes aspectos relacionados con la utilización de la aplicación para conseguir perfilar a los distintos alumnos, encontrar las causas de los problemas más comunes que se dan al manejar SRec y proporcionar las soluciones más adecuadas.

Además, se estudiarán diversas mejoras y ampliaciones técnicas, como por ejemplo en la interfaz de la aplicación para hacerla más intuitiva y acorde a las interfaces que se realizan en la actualidad y a la hora de manejar árboles grandes.

Capítulo 10. Conclusions

Along this PhD Thesis the work done by Antonio Pérez Carrasco has been exposed. After riding this work, one can see that this work has the common phases: identification of a problem to solve, review of the related state of the art, development of a software application to solve the problem, evaluation of that software application and actions for expanding the use of that software.

The Chapter 1 contains an introduction of the work done, providing the motivation, a start hypothesis, and some objectives that had to be covered. Main contributions are listed too. The Chapter 2 contains a complete state of the art; it reviews the current situation of the information visualization and the related researching areas with this work done: software visualization, programs visualization, and recursion visualization. Conceptual models about recursion have been explored and how they are used into the existing bibliography has been studied. Mental models have been reviewed too.

The Chapters 3 and 4 were focused on describing SRec from different points of view. First of these chapters gives a global view and talks about interface features, graphical representations generated by SRec, animation capacity and interaction options. The second chapter of them explores the implementation of SRec, reviewing the internal architecture for the generation of views; the data flow handled when SRec loads and processes a Java class, or how SRec uses XML language.

The Chapter 5 analyzes SRec effectiveness about visualization. On the one hand, the chapter makes a comparison between existing software focused on recursion visualization and SRec where readers can realize that SRec supports more conceptual models, more interaction possibilities and more animation controls. With this comparison, the contributions of SRec in a technical and educational level are brought up. On the other hand, the chapter has an analysis about SRec capacity for visualizing “divide-and-conquer” technique algorithms. A classification of “divide-and-conquer” algorithms was done for finding out algorithms that haven't a good representation by SRec. The objective was to have views for those kinds of algorithms improving the existing views or adding some new views. The needed changes were found, so the implementation is a future work.

The Chapters 6 and 7 contain the different usability evaluations made on SRec. In one hand, the Chapter 6 has the information about usability evaluations made with students in the last five academic years, providing numerical marks, suggestions and criticism. In each evaluation some conclusions were added. All data were collected by questionnaires. The Chapter 7 talks about usability, but it uses others methods. It exposes a brief study about how SRec is used (counting number of generated errors, size of the used examples, number of processed classes, etc.) and a summary about observations on users of SRec (what they make while they use SRec).

The Chapter 8 compiles taken actions to let SRec expand and be used in the classrooms. This Chapter collects different educational facilities provided by SRec. Besides, it talks about several aspects: the provided user documentation (manual and interactive help) and diffusion tasks as the capacity of SRec for working in several languages, the presence of SRec at some Conferences and Congresses and the developed plug-in for BlueJ.

The features of the implementation of SRec let it be extended easily for developers, and that make easier that other people (like students, for example) can learn the code quickly to improve or extend the application. Besides, XML usage can make easier an ulterior extension of SRec to be connected to others applications or, for example, start to generate SVG documents for viewing visualizations with a standard navigator application.

Usability evaluations have let to do a development of a tool very similar to the needed tool by students, but sometimes it had as a consequence that some numerical marks were too low. Nevertheless, students told spontaneously in the report of the practises they made that SRec helped them to do the tasks of understanding, analyzing and debugging. This is, definitively, the most important objective and it’s considered absolutely achieved and completed.

The set of evaluations on SRec will be very useful to improve the software tool in the future. The evaluation on how students use SRec and how they behaviour while they are using SRec will let research in the more problematic or worse-done parts of SRec to improve them and avoid users crash with them, making a smoother learning curve.

The evaluation for the adequacy of SRec for representing divide-and-conquer algorithms has let identify the more problematic kind of problems, so it will be possible to look for a representation model for that kind of problems in order to enlarge the set of problem that can be visualized with SRec, improving the current views or adding some new ones.

10.1 Satisfaction Hypothesis

After doing this work, the initial hypothesis is successfully covered. It said that making a software tool with an educational aim, interaction possibilities and an automatic way of use is possible. These three issues are developed next.

The built software tool has educational objectives. It can be used at Computer Science courses where recursion is tough (programming, algorithms design…). With visualizations, students can understand, analyze and debug recursive algorithms and divide-and-conquer algorithms in an easier way, according to them.

Besides, SRec has some interaction options for controlling the amount of information showed at the window, so user can increase the amount of information about the elements he sees, filter some data, explore the available elements, reconfigure the location of some elements, modify the format of the data, change the abstraction level, select the most appropriate view at each moment, make a step, make a jump, make an automatic animation, etc, These interaction options rise the engagement of students.

SRec has an automatic way of usage that allows students make a new visualization from scratch with very few mouse clicks, the needed ones to load a class, select a method and introduce the needed values. The creation and management of the visualizations elements is done by SRec, user remains totally free.

So, all these things confirm that the start hypothesis is totally satisfied; it’s possible to make a software application with an educational aim, interaction options and automatic way of usage.

10.2 Contributions

In a technical view of point, there are some contributions made thank to this work to the programs visualization paradigm.

The first contribution is the development of a software tool that contains some functionality and capacity not available so far for teaching community. SRec is not limited to a preset examples, it allows executing any recursive program designed by users. The creation of animations is absolutely automatic. They use until five views in divide-and-conquer case, this amount is not usual in this type of software.

SRec has some views never used so far in any software tool for visualizing programs or divide-and-conquer algorithms. Chronological view and Data structure view, focused on the data structure handled by the algorithm, are a very important differentiator element versus all the existing tools.

The handling of the visualization is very flexible with SRec; it increases highly the amount of possibilities while user is controlling the animation. Steps, jumps, advances to the end, and all of them in two possible directions (backwards and forwards) are the largest set of animation controls found ever in a programs visualization tool in an educational environment. SRec has a large set of interaction options (highlighting nodes, increasing of information about an element, filtering data, reconfiguring elements…) that are very numerous and very useful.

SRec has several teaching functions for helping teachers and students with their tasks. They are not included in most similar software; it makes SRec better than the rest existing tools. Besides, with SRec some installing facilities are provided, as auto run file or quick guides.

Along this working period, an exhaustive and long usability study has been done, so it has been possible to analyze the SRec acceptation evolution by students when the new functionalities were being added or improved. This study has let the application get better for students interests. The user centered design used here is not usual in visualization systems.

10.3 Future works

There are some works that have a technical character or researching point that will be done since 2012 more deeply and in a more independent way, but letting results and achievements of some of them make a positive influence on the rest. All of them are commented next.

One of the future works will be a study about the educational effectiveness of SRec. It hasn't been done along this work because it has a very technical character, focused on the development of a software tool following usability rules using a user centered design. However, it's very important to make a study about how SRec helps to students to learn and analyze algorithms. Now SRec is a stable and useful software tool, so it's a good moment to make this kind of studies.

A not-finished work is the integral study of the gathered information during the fifth usability evaluation:

- The way students use SRec.
- The way students behave when they are using SRec.
- Academic material (practise report).
- Questionnaire.

All these data will let establish user profiles, find causes for already found problems, discover how SRec has an influence on the work of students and improve the tool.

Prediction possibility can be a very important feature for SRec. So it's going to be studied for adding a new prediction mode [38] to SRec. It would let users write results and input values before SRec show them, and animations could confirm the correctness of the data written by the user. This would be a way to avoid paper usage while students work with SRec.

The internal architecture of SRec let generate an unlimited number of views depending on the kind of algorithm that is running. So, it's easy, and very interesting, to increase the number of design techniques that SRec can visualize. In this way, "dynamic programming" will be the next design technique that will be added to SRec after divide-and-conquer and recursion techniques. This will let SRec increase its functionality and

be used for a longer time in the courses, so students will be able to make more profitable the learning of SRec.

In the future, more views for representing of divide-and-conquer algorithms will be added to SRec. According to the made study, the geometrical problems are the kind of problems more difficult to visualize with SRec, so a new view for this kind of algorithms will be added. The view will be depending on the domain, but general enough to be able to visualize every geometrical algorithm. It would be good for higher level algorithms like multidimensional divide-and-conquer algorithms. This Thesis is already providing some proposals that will be improved and included in SRec in the future.

To learn more about how students use SRec will be very important for the future development of the tool and for making some student group according to how they use the application, how they behave when they are using SRec, etc. After that, it will be easier to find out the causes and solutions of the general problems of SRec usage.

Anexo 1: Descripción de los Documentos XML Empleados

Tal y como se menciona en el apartado “4.3 Utilización de XML“, SRec hace uso de diversos documentos XML. Se presenta a continuación la DTD (*Document Type Definition*) que define el contenido de los mismos.

A1.1 DTD de Documento XML para las Visualizaciones

```
<!ELEMENT Visualizacion (OpConf, OpFormato, DatosVisiblidad, Traza, TrazaCompleta)>

<!ELEMENT OpConf (DatosMostrar,MostrarHistoria,MostrarArbolSalto)>
<!ELEMENT DatosMostrar (EMPTY)>
<!ATTLIST DatosMostrar entrada (true|false) #REQUIRED>
<!ATTLIST DatosMostrar salida (true|false) #REQUIRED>
<!ELEMENT MostrarHistoria (EMPTY)>
<!ATTLIST MostrarHistoria estadoHistoria (Mantener|Atenuar|Eliminar) #REQUIRED>
<!ELEMENT MostrarArbolSalto (EMPTY)>
<!ATTLIST MostrarArbolSalto mostrarArbol (true|false) #REQUIRED>

<!ELEMENT OpFormato (Celda,Celda,Otros)>
<!ELEMENT Celda (Color,Color,Color,Color?)>
<!ATTLIST Celda degradado (true|false) #REQUIRED>
<!ATTLIST Celda nombre (entrada|salida) #REQUIRED>

<!ELEMENT                                                                    Otros
(Color,Color,Color,Color,Color,Color,Color,Color,Color,Color,Color,Color,Color,Color,Gro
sor,Grosor,Distancia,Distancia,Tipo,Tipo,Color,Color,Color,Color,Color,modoColor,modoCol
orDegr1,modoColorDegr2,Fuente,Fuente,FuenteTam,FuenteTam,Zoom,Zoom)>
<!ELEMENT Color (EMPTY)>
<!ATTLIST Color r CDATA #REQUIRED>
<!ATTLIST Color g CDATA #REQUIRED>
<!ATTLIST Color b CDATA #REQUIRED>
<!ATTLIST Color destino
(fuente|color1|color2|color1a|color2a|color1nc|color2nc|marcoActual|caminoActual|panel|f
lechacodigoPR|codigoCo|codigoMF|codigoMB|codigoRC|codigoFP|trazaE|trazaS|trazaFP)
#REQUIRED>
<!ELEMENT Grosor (EMPTY)>
<!ATTLIST Grosor destino (flecha|marcoActual) #REQUIRED>
<!ATTLIST Grosor tam (CDATA) #REQUIRED>
<!ELEMENT Distancia (EMPTY)>
<!ATTLIST Distancia destino (vertical|horizontal) #REQUIRED>
<!ATTLIST Distancia tam (CDATA) #REQUIRED>
<!ELEMENT Tipo (EMPTY)>
<!ATTLIST Tipo destino (bordeCelda|formaFlecha) #REQUIRED>
<!ATTLIST Tipo tam (CDATA) #REQUIRED>
<!ELEMENT modoColor (EMPTY)>
<!ATTLIST modoColor destino (modo) #REQUIRED>
<!ATTLIST modoColor tam (CDATA) #REQUIRED>
<!ELEMENT modoColorDegr1 (EMPTY)>
<!ATTLIST modoColorDegr1 destino (degr) #REQUIRED>
<!ATTLIST modoColorDegr1 tam (true|false) #REQUIRED>
<!ELEMENT modoColorDegr2 (EMPTY)>
<!ATTLIST modoColorDegr2 destino (degr) #REQUIRED>
<!ATTLIST modoColorDegr2 tam (true|false) #REQUIRED>
<!ELEMENT Fuente (EMPTY)>
<!ATTLIST Fuente destino (trazaE|trazaS) #REQUIRED>
<!ATTLIST Fuente tam (CDATA) #REQUIRED>
<!ELEMENT FuenteTam (EMPTY)>
<!ATTLIST FuenteTam destino (trazaE|trazaS) #REQUIRED>
<!ATTLIST FuenteTam tam (CDATA) #REQUIRED>
<!ELEMENT Zoom (EMPTY)>
<!ATTLIST Zoom destino (arbol|pila) #REQUIRED>
<!ATTLIST Zoom tam (CDATA) #REQUIRED>

<!ELEMENT DatosVisibilidad (Metodo+)>
```

```
<!ELEMENT Metodo (Param)>
<!ATTLIST Metodo metodoPrincipal (true|false) #REQUIRED>
<!ATTLIST Metodo metodoVisible (true|false) #REQUIRED>
<!ATTLIST Metodo nombre (CDATA) #REQUIRED>
<!ATTLIST Metodo retorno (true|false) #REQUIRED>
<!ELEMENT Param (ParamE+,ParamS+)>
<!ELEMENT ParamE (EMPTY)>
<!ATTLIST ParamE dim (CDATA) #REQUIRED>
<!ATTLIST ParamE nombre (CDATA) #REQUIRED>
<!ATTLIST ParamE orden (CDATA) #REQUIRED>
<!ATTLIST ParamE tipo (byte|short|int|long|float|double|boolean|String|char) #REQUIRED>
<!ATTLIST ParamE visible (true|false) #REQUIRED>
<!ELEMENT ParamS (EMPTY)>
<!ATTLIST ParamS dim (CDATA) #REQUIRED>
<!ATTLIST ParamS nombre (CDATA) #REQUIRED>
<!ATTLIST ParamS orden (CDATA) #REQUIRED>
<!ATTLIST ParamS tipo (byte|short|int|long|float|double|boolean|String|char) #REQUIRED>
<!ATTLIST ParamS visible (true|false) #REQUIRED>



<!ELEMENT Traza (Datos,RegistroActivacion)>

<!ELEMENT Datos (EMPTY)>

<!ATTLIST Datos archivo CDATA #REQUIRED>
<!ATTLIST Datos idTraza CDATA #REQUIRED>
<!ATTLIST Datos metodoEjecucion CDATA #REQUIRED>
<!ATTLIST Datos nombre CDATA #REQUIRED>

<!ELEMENT RegistroActivacion (Valor,Param,Metodo,Hijos)>

<!ELEMENT Valor (EMPTY)>
<!ATTLIST Valor actual (true|false) #REQUIRED>
<!ATTLIST Valor caminoActual (true|false) #REQUIRED>
<!ATTLIST Valor dimE1 (CDATA) #REQUIRED>
<!ATTLIST Valor dimE2 (CDATA) #IMPLIED>
<!ATTLIST Valor dimE3 (CDATA) #IMPLIED>
<!ATTLIST Valor dimE4 (CDATA) #IMPLIED>
<!ATTLIST Valor dimE5 (CDATA) #IMPLIED>
<!ATTLIST Valor dimS1 (CDATA) #REQUIRED>
<!ATTLIST Valor dimS2 (CDATA) #IMPLIED>
<!ATTLIST Valor dimS3 (CDATA) #IMPLIED>
<!ATTLIST Valor dimS4 (CDATA) #IMPLIED>
<!ATTLIST Valor dimS5 (CDATA) #IMPLIED>
<!ATTLIST Valor entradaVisible (true|false) #REQUIRED>
<!ATTLIST Valor paramE1 (CDATA) #REQUIRED>
<!ATTLIST Valor paramE2 (CDATA) #IMPLIED>
<!ATTLIST Valor paramE3 (CDATA) #IMPLIED>
<!ATTLIST Valor paramE4 (CDATA) #IMPLIED>
<!ATTLIST Valor paramE5 (CDATA) #IMPLIED>
<!ATTLIST Valor paramS1 (CDATA) #REQUIRED>
<!ATTLIST Valor paramS2 (CDATA) #IMPLIED>
<!ATTLIST Valor paramS3 (CDATA) #IMPLIED>
<!ATTLIST Valor paramS4 (CDATA) #IMPLIED>
<!ATTLIST Valor paramS5 (CDATA) #IMPLIED>
<!ATTLIST Valor salidaVisible (true|false) #REQUIRED>
<!ATTLIST Valor tipoE1 (CDATA) #REQUIRED>
<!ATTLIST Valor tipoE2 (CDATA) #IMPLIED>
<!ATTLIST Valor tipoE3 (CDATA) #IMPLIED>
<!ATTLIST Valor tipoE4 (CDATA) #IMPLIED>
<!ATTLIST Valor tipoE5 (CDATA) #IMPLIED>
<!ATTLIST Valor tipoS1 (CDATA) #REQUIRED>
<!ATTLIST Valor tipoS2 (CDATA) #IMPLIED>
<!ATTLIST Valor tipoS3 (CDATA) #IMPLIED>
<!ATTLIST Valor tipoS4 (CDATA) #IMPLIED>
<!ATTLIST Valor tipoS5 (CDATA) #IMPLIED>

<!ELEMENT Param (EMPTY)>
<!ATTLIST Param contraido (true|false) #REQUIRED>
<!ATTLIST Param hijoVisible (CDATA) #REQUIRED>
<!ATTLIST Param historico (CDATA) #REQUIRED>
<!ATTLIST Param iluminado (CDATA) #REQUIRED>
<!ATTLIST Valor inhibido (true|false) #REQUIRED>
<!ATTLIST Valor mostradoEntero (true|false) #REQUIRED>
<!ATTLIST Param numID (CDATA) #REQUIRED>
<!ATTLIST Param numHijos (CDATA) #REQUIRED>
```

```
<!ELEMENT Metodo (EMPTY)>
<!ATTLIST Metodo devuelveValor (true|false) #REQUIRED>
<!ATTLIST Metodo nombreMetodo (true|false) #REQUIRED>
<!ATTLIST Metodo nombreParametro1 (CDATA) #REQUIRED>
<!ATTLIST Metodo nombreParametro2 (CDATA) #IMPLIED>
<!ATTLIST Metodo nombreParametro3 (CDATA) #IMPLIED>
<!ATTLIST Metodo nombreParametro4 (CDATA) #IMPLIED>
<!ATTLIST Metodo nombreParametro5 (CDATA) #IMPLIED>

       <!ELEMENT Hijos (RegistroActivacion)*>
```

A1.2 DTD de Documento XML para las Opciones de Configuración

```
<!ELEMENT Opciones (OpcionIdioma, OpcionOpsVisualizacion, OpcionConfVisualizacion,
OpcionTipoGrafico, OpcionMVJava, OpcionBorradoFicheros, OpcionFicherosRecientes)>

<!ELEMENT OpcionIdioma (idioma)>
<!ELEMENT idioma (EMPTY)>
<!ATTLIST idioma valor CDATA #REQUIRED>

<!ELEMENT OpcionOpsVisualizacion (historia,datosMostrar,mostrarArbolSalto,mostrarVisor)>
<!ELEMENT historia (EMPTY)>
<!ATTLIST historia valor (Atenuar|Mantener|Eliminar)>
<!ELEMENT datosMostrar (EMPTY)>
<!ATTLIST datosMostrar entrada (true|false)>
<!ATTLIST datosMostrar salida (true|false)>
<!ELEMENT mostrarArbolSalto (EMPTY)>
<!ATTLIST mostrarArbolSalto valor (true|false)>
<!ELEMENT mostrarVisor (EMPTY)>
<!ATTLIST mostrarVisor valor (true|false)>

<!ELEMENT OpcionConfVisualizacion
(colorFEntrada,colorFSalida,colorC1Entrada,colorC1Salida,
colorC1AEntrada,colorC1ASalida,colorC1NCSalida,degradados,modoColor,colorFlecha,colorPan
el,colorActual,colorCActual,
varios,colorPalabrasReservadas,colorComentarios,colorMetodoForeground,colorMetodoBackgro
und,
colorCodigo,colorIluminado,fuentesTrazaCodigo,zoomsDefecto,distancias,colorm2_0,colorm2_
1,colorm2_2,colorm2_3,
colorm2_4,colorm2_5,colorm2_6,colorm2_7,colorm2_8,colorm2_9)>
<!ELEMENT colorFEntrada (EMPTY)>
<!ATTLIST colorFEntrada r CDATA #REQUIRED>
<!ATTLIST colorFEntrada g CDATA #REQUIRED>
<!ATTLIST colorFEntrada b CDATA #REQUIRED>
<!ELEMENT colorFSalida (EMPTY)>
<!ATTLIST colorFSalida r CDATA #REQUIRED>
<!ATTLIST colorFSalida g CDATA #REQUIRED>
<!ATTLIST colorFSalida b CDATA #REQUIRED>
<!ELEMENT colorC1Entrada (EMPTY)>
<!ATTLIST colorC1Entrada r CDATA #REQUIRED>
<!ATTLIST colorC1Entrada g CDATA #REQUIRED>
<!ATTLIST colorC1Entrada b CDATA #REQUIRED>
<!ELEMENT colorC1Salida (EMPTY)>
<!ATTLIST colorC1Salida r CDATA #REQUIRED>
<!ATTLIST colorC1Salida g CDATA #REQUIRED>
<!ATTLIST colorC1Salida b CDATA #REQUIRED>
<!ELEMENT colorC1AEntrada (EMPTY)>
<!ATTLIST colorC1AEntrada r CDATA #REQUIRED>
<!ATTLIST colorC1AEntrada g CDATA #REQUIRED>
<!ATTLIST colorC1AEntrada b CDATA #REQUIRED>
<!ELEMENT colorC1ASalida (EMPTY)>
<!ATTLIST colorC1ASalida r CDATA #REQUIRED>
<!ATTLIST colorC1ASalida g CDATA #REQUIRED>
<!ATTLIST colorC1ASalida b CDATA #REQUIRED>
<!ELEMENT colorC1NCSalida (EMPTY)>
<!ATTLIST colorC1NCSalida r CDATA #REQUIRED>
<!ATTLIST colorC1NCSalida g CDATA #REQUIRED>
<!ATTLIST colorC1NCSalida b CDATA #REQUIRED>
<!ELEMENT degradados (EMPTY)>
<!ATTLIST degradados modo1 (true|false) #REQUIRED>
<!ATTLIST degradados modo2 (true|false) #REQUIRED>
```

```
<!ELEMENT modoColor (EMPTY)>
<!ATTLIST modoColor modo (1|2) #REQUIRED>
<!ELEMENT colorFlecha (EMPTY)>
<!ATTLIST colorFlecha r CDATA #REQUIRED>
<!ATTLIST colorFlecha g CDATA #REQUIRED>
<!ATTLIST colorFlecha b CDATA #REQUIRED>
<!ELEMENT colorPanel (EMPTY)>
<!ATTLIST colorPanel r CDATA #REQUIRED>
<!ATTLIST colorPanel g CDATA #REQUIRED>
<!ATTLIST colorPanel b CDATA #REQUIRED>
<!ELEMENT colorActual (EMPTY)>
<!ATTLIST colorActual r CDATA #REQUIRED>
<!ATTLIST colorActual g CDATA #REQUIRED>
<!ATTLIST colorActual b CDATA #REQUIRED>
<!ELEMENT colorCActual (EMPTY)>
<!ATTLIST colorCActual r CDATA #REQUIRED>
<!ATTLIST colorCActual g CDATA #REQUIRED>
<!ATTLIST colorCActual b CDATA #REQUIRED>
<!ELEMENT varios (EMPTY)>
<!ATTLIST varios grosorFlecha CDATA #REQUIRED>
<!ATTLIST varios grosorMarcos CDATA #REQUIRED>
<!ATTLIST varios tipoBordeCelda CDATA #REQUIRED>
<!ATTLIST varios tipoFlecha CDATA #REQUIRED>
<!ELEMENT colorPalabrasReservadas (EMPTY)>
<!ATTLIST colorPalabrasReservadas r CDATA #REQUIRED>
<!ATTLIST colorPalabrasReservadas g CDATA #REQUIRED>
<!ATTLIST colorPalabrasReservadas b CDATA #REQUIRED>
<!ELEMENT colorComentarios (EMPTY)>
<!ATTLIST colorComentarios r CDATA #REQUIRED>
<!ATTLIST colorComentarios g CDATA #REQUIRED>
<!ATTLIST colorComentarios b CDATA #REQUIRED>
<!ELEMENT colorMetodoForeground (EMPTY)>
<!ATTLIST colorMetodoForeground r CDATA #REQUIRED>
<!ATTLIST colorMetodoForeground g CDATA #REQUIRED>
<!ATTLIST colorMetodoForeground b CDATA #REQUIRED>
<!ELEMENT colorMetodoBackground (EMPTY)>
<!ATTLIST colorMetodoBackground r CDATA #REQUIRED>
<!ATTLIST colorMetodoBackground g CDATA #REQUIRED>
<!ATTLIST colorMetodoBackground b CDATA #REQUIRED>
<!ELEMENT colorCodigo (EMPTY)>
<!ATTLIST colorCodigo r CDATA #REQUIRED>
<!ATTLIST colorCodigo g CDATA #REQUIRED>
<!ATTLIST colorCodigo b CDATA #REQUIRED>
<!ELEMENT colorIluminado (EMPTY)>
<!ATTLIST colorIluminado r CDATA #REQUIRED>
<!ATTLIST colorIluminado g CDATA #REQUIRED>
<!ATTLIST colorIluminado b CDATA #REQUIRED>
<!ELEMENT fuentesTrazaCodigo (EMPTY)>
<!ATTLIST fuentesTrazaCodigo fuenteCodigo CDATA #REQUIRED>
<!ATTLIST fuentesTrazaCodigo fuenteTraza CDATA #REQUIRED>
<!ATTLIST fuentesTrazaCodigo tamFuenteCodigo CDATA #REQUIRED>
<!ATTLIST fuentesTrazaCodigo tamFuenteTraza CDATA #REQUIRED>
<!ELEMENT zoomsDefecto (EMPTY)>
<!ATTLIST zoomsDefecto zoomArbol CDATA #REQUIRED>
<!ATTLIST zoomsDefecto zoomPila CDATA #REQUIRED>
<!ELEMENT distancias (EMPTY)>
<!ATTLIST distancias horizontal CDATA #REQUIRED>
<!ATTLIST distancias vertical CDATA #REQUIRED>
<!ELEMENT colorm2_0 (EMPTY)>
<!ATTLIST colorm2_0 r CDATA #REQUIRED>
<!ATTLIST colorm2_0 g CDATA #REQUIRED>
<!ATTLIST colorm2_0 b CDATA #REQUIRED>
<!ELEMENT colorm2_1 (EMPTY)>
<!ATTLIST colorm2_1 r CDATA #REQUIRED>
<!ATTLIST colorm2_1 g CDATA #REQUIRED>
<!ATTLIST colorm2_1 b CDATA #REQUIRED>
<!ELEMENT colorm2_2 (EMPTY)>
<!ATTLIST colorm2_2 r CDATA #REQUIRED>
<!ATTLIST colorm2_2 g CDATA #REQUIRED>
<!ATTLIST colorm2_2 b CDATA #REQUIRED>
<!ELEMENT colorm2_3 (EMPTY)>
<!ATTLIST colorm2_3 r CDATA #REQUIRED>
<!ATTLIST colorm2_3 g CDATA #REQUIRED>
<!ATTLIST colorm2_3 b CDATA #REQUIRED>
<!ELEMENT colorm2_4 (EMPTY)>
<!ATTLIST colorm2_4 r CDATA #REQUIRED>
<!ATTLIST colorm2_4 g CDATA #REQUIRED>
```

```
<!ATTLIST colorm2_4 b CDATA #REQUIRED>
<!ELEMENT colorm2_5 (EMPTY)>
<!ATTLIST colorm2_5 r CDATA #REQUIRED>
<!ATTLIST colorm2_5 g CDATA #REQUIRED>
<!ATTLIST colorm2_5 b CDATA #REQUIRED>
<!ELEMENT colorm2_6 (EMPTY)>
<!ATTLIST colorm2_6 r CDATA #REQUIRED>
<!ATTLIST colorm2_6 g CDATA #REQUIRED>
<!ATTLIST colorm2_6 b CDATA #REQUIRED>
<!ELEMENT colorm2_7 (EMPTY)>
<!ATTLIST colorm2_7 r CDATA #REQUIRED>
<!ATTLIST colorm2_7 g CDATA #REQUIRED>
<!ATTLIST colorm2_7 b CDATA #REQUIRED>
<!ELEMENT colorm2_8 (EMPTY)>
<!ATTLIST colorm2_8 r CDATA #REQUIRED>
<!ATTLIST colorm2_8 g CDATA #REQUIRED>
<!ATTLIST colorm2_8 b CDATA #REQUIRED>
<!ELEMENT colorm2_9 (EMPTY)>
<!ATTLIST colorm2_9 r CDATA #REQUIRED>
<!ATTLIST colorm2_9 g CDATA #REQUIRED>
<!ATTLIST colorm2_9 b CDATA #REQUIRED>

<!ELEMENT OpcionTipoGrafico (tipo,tipo,tipo)>
<!ELEMENT Tipo (EMPTY)>
<!ATTLIST Tipo nombre (JPEG|GIF|PNG)>
<!ATTLIST Tipo ultimavez (true|false)>

<!ELEMENT OpcionMVJava (dir,version,valida)>
<!ELEMENT dir (EMPTY)>
<!ATTLIST dir valor CDATA #REQUIRED>
<!ELEMENT version (EMPTY)>
<!ATTLIST version valor CDATA #REQUIRED>
<!ELEMENT valida (EMPTY)>
<!ATTLIST valida valor (true|false) #REQUIRED>

<!ELEMENT OpcionBorradoFicheros
(xml_original,xml_generado,java_generado,class_original,class_generado)>
<!ELEMENT xml_original (EMPTY)>
<!ATTLIST xml_original valor (trae|false) #REQUIRED>
<!ELEMENT xml_generado (EMPTY)>
<!ATTLIST xml_ generado valor (trae|false) #REQUIRED>
<!ELEMENT java_generado (EMPTY)>
<!ATTLIST java_generado valor (trae|false) #REQUIRED>
<!ELEMENT class_original (EMPTY)>
<!ATTLIST class_original valor (trae|false) #REQUIRED>
<!ELEMENT class_generado (EMPTY)>
<!ATTLIST class_generado valor (trae|false) #REQUIRED>

<!ELEMENT OpcionFicherosRecientes (dir)>
```

A1.3 DTD de Documento XML para los Parámetros de un Método

```
<!ELEMENT valoresAlgoritmoVisualizador (algoritmo)>
<!ELEMENT algoritmo (parámetro+)>
<!ATTLIST algoritmo nombre CDATA #REQUIRED>
<!ELEMENT parametro (EMPTY)>
<!ATTLIST parametro val CDATA #REQUIRED>
<!ATTLIST parametro clase CDATA #REQUIRED>
```

A1.4 DTD de Documento XML para la Traducción de Código Java

Este documento XML se genera con el paquete externo ToXML (programado por Harsh Jain [23]), motivo por el cual no se proporcionará su DTD, dada además la alta complejidad de la misma.

Anexo 2: Enunciados de las Prácticas y Cuestionarios

En este apartado se exponen íntegramente los enunciados de las prácticas que realizaron los alumnos durante las diferentes evaluaciones de SRec. Los enunciados que se presentan mantienen también todos los ejercicios de introducción realizados tanto por el profesor como por los propios alumnos en las distintas sesiones en los laboratorios.

Por otra parte, también se incluyen en este anexo los cuestionarios que tuvieron que responder los alumnos en cada una de las evaluaciones de SRec. Tal y como se explicó en el apartado “6.3 Método General“, el último de ellos se realizó de manera electrónica, de ahí que presente un formato claramente distinto a los anteriores, si bien sus preguntas son muy similares.

A2.1 Primera Evaluación de SRec

Sesión de Evaluación de SRec

SRec es una aplicación para la visualización y animación de métodos recursivos. Esta sesión de laboratorio se realiza para evaluar la calidad de SRec. La sesión consta de varias fases a realizar secuencialmente, que se describen a continuación.

1. **Demostración del profesor.** Duración: 10-15 minutos. Algoritmo: serie de Fibonacci recursiva.

```
public static int fib (int n) {
   if (n==0 || n==1)
      return 1;
   else
      return fib(n-1) + fib(n-2);
}
```

   El profesor realizará una demostración del funcionamiento de SRec en la que mostrará el funcionamiento de sus funciones principales: abrir ficheros, almacenar/cargar una animación, reproducir la animación, manejar paneles y cambiar opciones de configuración.

2. **Primera animación.** Duración aproximada: 20 minutos. Algoritmo: exponenciación mediante partición binaria del exponente.

```
public static int pot (int b, int e) {
   if (e==0)
      return 1;
   else if (e%2==0)
      return pot(b*b,e/2);
   else
      return b*pot(b*b,e/2);
}
```

   El alumno generará una animación de este algoritmo. Para ello, debe cargar la clase *ClaseEvaluacion* disponible en la Web de la asignatura, generar una animación, almacenarla y cargarla de nuevo.

3. **Segunda animación.** Duración aproximada: 20 minutos. Algoritmo: números combinatorios.

```
public static int comb (int m, int n) {
   if (n==0)
      return 1;
   else if (m==n)
      return 1;
   else
      return comb(m-1,n) + comb(m-1,n-1);
 }
```

De nuevo, el alumno experimentará con una animación de un algoritmo recursivo múltiple. Se le pide que genere una animación y, después, experimente con las múltiples vistas y con las opciones de configuración.

4. **Tercera animación.** Duración aproximada: 40 minutos. Algoritmo: ordenación por mezcla.

```
private static void ordenarPorMezcla2 (int[] v, int inf, int sup) {
   if (inf<sup) {
      int medio = (inf+sup)/2;
      ordenarPorMezcla2 (v, inf, medio);
      ordenarPorMezcla2 (v, medio, sup);
      mezclar2 (v, inf, medio, sup);
   }
}
public static void mezclar2 (int[] v, int inf, int medio, int sup){
   int[] vAux = new int[sup-inf+1]; //vector auxiliar de igual long
   int i1 = inf;
   int i2 = medio+1;
```

```
        int j = 0; //inicialización de índice del vector auxiliar
        while (i1<=medio && i2<=sup) {
           if (v[i1]>v[i2]) {
              vAux[j] = v[i1];
              i1++;
           }
           else {
              vAux[j] = v[i2];
              i2++;
           }
           j++;
        }
        for (int i=i1; i<=medio; i++) {
           vAux[j] = v[i];
           j++;
        }
        for (int i=i2; i<=sup; i++) {
           vAux[j] = v[i];
           j++;
        }
        for (int i=inf; i<=sup; i++)
           v[i] = vAux[i-inf]; //transformación de índices y copia
     }
```

El alumno debe depurar este programa que implementa el algoritmo de ordenación por mezcla. Puede servirse de SRec o estudiar el código de forma estática. Debe entregar un fichero (nombrado con los apellidos del alumno) en el que especifique los números de línea que contienen errores, cada fragmento de código erróneo y cada nuevo fragmento.

5. **Cuestionario.** Duración aproximada: 20 minutos.
   El alumno debe responder a las preguntas siguientes.

Cuestionario

En las dos preguntas siguientes, marca la opción de cada pregunta con la que estás de acuerdo.

1. Sobre la facilidad de uso de SRec:
   [ ] La herramienta es fácil de usar
   [ ] La herramienta es fácil de usar en unas partes pero difícil en otras. Identifique qué partes: _____
   [ ] La herramienta es difícil de usar
2. Sobre la utilidad de SRec:
   [ ] La herramienta es muy o bastante útil
   [ ] La herramienta es algo útil
   [ ] La herramienta es poco o nada útil

En las tres preguntas siguientes, evalúa cada opción con un número comprendido entre 1 (muy malo) y 5 (muy bueno).

3. Evalúa la calidad de las siguientes características de SRec:
   [ ] Estructura del menú principal
   [ ] Facilidades de almacenar/cargar animación
   [ ] Facilidades de configuración de las visualizaciones
   [ ] Controles de animación
   [ ] Múltiples vistas de la recursividad
4. Evalúa la utilidad de las siguientes características de SRec:
   [ ] Facilidades de almacenar/cargar animación
   [ ] Facilidades de configuración de las visualizaciones
   [ ] Controles de animación
   [ ] Múltiples vistas de la recursividad
5. Evalúa la utilidad de las siguientes representaciones de la recursividad:
   [ ] Traza
   [ ] Pila de control
   [ ] Árbol de activación
6. Evalúa la utilidad de SRec para depurar el algoritmo de ordenación por mezcla: _____
7. Evalúa la utilidad de las siguientes representaciones de la recursividad para depurar:
   [ ] Traza
   [ ] Pila de control
   [ ] Árbol de activación
8. Evalúa cuánto te ha gustado SRec: _____

Responde a las siguientes preguntas en formato libre:

9. Di qué características te parece que serían útiles pero SRec carece de ellas:
   _____
   _____
   _____

10. Di qué características de SRec te parecen tan poco útiles que las suprimirías:
    _____
    _____
    _____

11. Describe las ventajas que encuentras en SRec:
    _____
    _____
    _____

12. Describe los inconvenientes que encuentras en SRec:
    _____
    _____
    _____

A2.2 Segunda Evaluación de SRec

Ingeniería Informática
Asignatura *Diseño y Análisis de Algoritmos*

Curso 2007/2008

Práctica nº 2

Objetivo

El objetivo de la práctica es que el alumno practique en la eliminación de la recursividad múltiple.

Carácter

La práctica es voluntaria. La sesión en el laboratorio se realizará individualmente. El resto de la práctica puede realizarse en parejas.

Prerrequisitos

El alumno debe tener nociones básicas de eliminación de la recursividad múltiple.

Enunciado

SRec es una aplicación para la visualización y animación de métodos recursivos. La práctica consta de dos partes, una a realizar en el laboratorio y otra, fuera del mismo.

La sesión de laboratorio también sirve para evaluar la calidad de SRec. La sesión consta de cuatro fases a realizar secuencialmente:

1. **Demostración del profesor.** Duración aproximada: 15 minutos. Algoritmo: serie de Fibonacci recursiva.

```
public static int fib (int n) {
   if (n==0 || n==1)
      return 1;
   else
      return fib(n-1) + fib(n-2);
}
```

El profesor realizará una demostración del funcionamiento de SRec en la que mostrará, mediante este algoritmo recursivo múltiple, el funcionamiento de sus funciones principales: seleccionar una máquina virtual de Java, procesar un fichero, generar y reproducir una animación, manejar paneles, almacenar y cargar la animación, y cambiar las diversas opciones de configuración.

2. **Primera animación.** Duración aproximada: 15 minutos. Algoritmo: exponenciación mediante partición binaria del exponente.

```
public static int pot (int b, int e) {
      if (e==0)
         return 1;
      else if (e%2==0)
         return pot(b*b,e/2);
      else
         return b*pot(b*b,e/2);
}
```

El alumno generará una animación de este algoritmo recursivo lineal. Para ello, debe realizar las siguientes tareas:

- Procesar la clase *ClaseEvaluacion* disponible en el sitio *web* de la asignatura (menú de Archivo).
- Generar una animación (menú de Archivo).
- Experimentar con todos los controles de animación hasta que se comprenda su funcionamiento.
- Almacenar la animación y cargarla de nuevo (ambas operaciones, en menú de Archivo).

3. **Segunda animación.** Duración aproximada: 20 minutos. Algoritmo: números combinatorios.

```
public static int comb (int m, int n) {      if (n==0)
      return 1;
   else if (m==n)
      return 1;
   else
      return comb(m-1,n) + comb(m-1,n-1);
}
```

El alumno experimentará con una animación de este algoritmo recursivo múltiple. Esta vez se le pide que realice las siguientes tareas:

- Generar una animación.
- Experimentar con los controles de animación (parte superior derecha de la ventana), observando el efecto de cada operación sobre cada una de las vistas de la recursividad:
    1. Traza.
    2. Pila de control.
    3. Árbol de activación.
- Experimentar con las opciones de configuración (menú Configuración) de:

1. Control de la cantidad de información mostrada en cada nodo del árbol.
2. Control de la visualización de la historia pasada.
3. Control del formato gráfico de las 3 visualizaciones (traza, pila de control, árbol de activación).

4. **Tercera animación.** Duración aproximada: 40 minutos. Problema: competición.

Dos participantes A y B juegan una competición que es ganada por el primero que venza en *n* partidos, *n*>0. En principio, ambos participantes tienen cualidades y preparación similares, por lo que cada uno tiene un 50% de probabilidad de ganar cada partido. Se quiere conocer la probabilidad que tiene el equipo A de ganar la competición si se sabe que A necesita *i* partidos para ganar y B necesita *j*. Obsérvese que al menos *i*>0 ó *j*>0 para que la situación tenga sentido.

Podemos denotar el problema como *prob*(*i*,*j*). Una solución directa es la siguiente:

```
public static float prob (int i, int j) {
   if (i==0)
      return (float)1.0;
   else if (j==0)
      return (float)0.0;
   else
      return (float)((prob(i,j-1)+prob(i-1,j))/2.0);
}
```

Se pretende eliminar la redundancia existente en este algoritmo recursivo múltiple. Durante el resto de la sesión se pide desarrollar un árbol de recursión y su grafo de activación asociado, que sean representativos del algoritmo. Puede servirse de SRec o estudiar el código de forma estática.

Entrega

Ambas representaciones gráficas se entregarán al profesor de la asignatura en una hoja (con los apellidos del alumno).

Posteriormente, estas representaciones se usarán como base para aplicar las técnicas de memorización y de tabulación. El plazo de entrega de la práctica completa es el 10 de diciembre de 2007. Debe enviarse por correo electrónico, adjuntando un breve informe siguiendo el modelo disponible en el sitio *Web* de la asignatura.

5. **Cuestionario.** Duración aproximada: 20 minutos.

El alumno debe responder a las preguntas siguientes.

Cuestionario de opinión sobre el sistema SRec

Nombre y apellidos (opcional): _____

En las preguntas siguientes, marca un valor en cada pregunta. Debes usar un valor de la escala mostrada en la siguiente tabla. Según la clase de pregunta, su significado se referirá a opinión o calidad:

Valor	Opinión	Calidad
1	Nada de acuerdo	Muy mala
2	Poco de acuerdo	Mala
3	Sin opinión	Regular
4	Algo de acuerdo	Buena
5	Totalmente de acuerdo	Muy buena

Te parece que SRec es **fácil de usar**: **[ ]**

Las partes que te parecen más **difíciles de usar** (si las hay) son:

_____
_____

Te parece que SRec **te ha ayudado** a analizar los algoritmos recursivos para:

[ ] Analizar qué llamadas se realizan en tiempo de ejecución
[ ] Identificar las dependencias entre llamadas

Te parece que, **la calidad en general** de SRec para analizar la recursividad es alta: **[ ]**

Las partes de **mejor calidad**, para ti, son:

_____
_____

Las partes de peor calidad, para ti son:

_____
_____

Te parece que **la calidad de varias partes** de SRec es alta:

[ ] Estructura del menú principal
[ ] Iconos
[ ] Controles de animación
[ ] Vista de traza
[ ] Vista de pila de control
[ ] Vista de árbol de activación
[ ] Configuración de las visualizaciones
[ ] Interacción con los paneles (*scroll*, mover paneles, *zoom*)

En conjunto, **te ha gustado** SRec: **[ ]**

Responde a las siguientes preguntas en formato libre:

1. Di qué características te parece que podrían ser **útiles** pero SRec carece de ellas:
   _____
   _____
   _____

2. Di qué características de SRec te parecen tan **poco útiles** que las suprimirías:
   _____
   _____
   _____

3. Describe los **aspectos positivos** que encuentras en SRec (sobre todo si no se han mencionado antes):
   _____
   _____
   _____

4. Describe los **aspectos negativos** que encuentras en SRec (sobre todo si no se han mencionado antes)
   _____
   _____
   _____

A2.3 Tercera Evaluación de SRec

Ingeniería Informática
Asignatura *Diseño y Análisis de Algoritmos*

Curso 2008/2009
Práctica nº 2

Objetivo

El objetivo de la práctica es que el alumno practique en la eliminación de la recursividad múltiple.

Carácter

La práctica es voluntaria. La sesión en el laboratorio se realizará individualmente. El resto de la práctica puede realizarse en parejas.

Prerrequisitos

El alumno debe tener nociones básicas de eliminación de la recursividad múltiple.

Enunciado

SRec es una aplicación para la visualización y animación de métodos recursivos. La práctica consta de dos partes, una a realizar en el laboratorio y otra, fuera del mismo.

La sesión de laboratorio también sirve para evaluar la calidad de SRec. La sesión consta de cuatro fases a realizar secuencialmente:

1. **Demostración del profesor.** Duración aproximada: 15 minutos. Algoritmo: serie de Fibonacci recursiva.

```
public static int fib (int n) {
   if (n==0 || n==1)
      return 1;
   else
      return fib(n-1) + fib(n-2);
}
```

   El profesor realizará una demostración del funcionamiento de SRec en la que mostrará, mediante este algoritmo recursivo múltiple, el funcionamiento de sus funciones principales: seleccionar una máquina virtual de Java, procesar un fichero, generar y reproducir una animación, manejar paneles, almacenar y cargar la animación, y cambiar las diversas opciones de configuración.

2. **Primera animación.** Duración aproximada: 15 minutos. Algoritmo: exponenciación mediante partición binaria del exponente.

```
public static int pot (int b, int e) {
   if (e==0)
      return 1;
   else if (e%2==0)
      return pot(b*b,e/2);
   else
      return b*pot(b*b,e/2);
}
```

El alumno generará una animación de este algoritmo recursivo lineal. Para ello, debe realizar las siguientes tareas:

- Procesar la clase *ClaseEvaluacion* disponible en el sitio *Web* de la asignatura (menú de Archivo).
- Generar una animación (menú de Archivo).
- Experimentar con todos los controles de animación hasta que se comprenda su funcionamiento.
- Almacenar la animación y cargarla de nuevo (ambas operaciones, en menú de Archivo).

3. **Segunda animación.** Duración aproximada: 20 minutos. Algoritmo: números combinatorios.

```
public static int comb (int m, int n) {
   if (n==0)
      return 1;
   else if (m==n)
      return 1;
   else
      return comb(m-1,n) + comb(m-1,n-1);
}
```

El alumno experimentará con una animación de este algoritmo recursivo múltiple. Esta vez se le pide que realice las siguientes tareas:

- Generar una animación.
- Experimentar con los controles de animación (parte superior derecha de la ventana), observando el efecto de cada operación sobre cada una de las vistas de la recursividad:
    1. Traza.
    2. Pila de control.
    3. Árbol de activación.
- Experimentar con las opciones de configuración (menú Configuración) de:
    1. Control de la cantidad de información mostrada en cada nodo del árbol.
    2. Control de la visualización de la historia pasada.
    3. Control del formato gráfico de las 3 visualizaciones (traza, pila de control, árbol de activación).

4. **Tercera animación.** Duración aproximada: 40 minutos. Algoritmo: competición.

Dos participantes A y B juegan una competición que es ganada por el primero que venza en *n* partidos, *n*>0. En principio, ambos participantes tienen cualidades y preparación similares, por lo que cada uno tiene un 50% de probabilidad de ganar cada partido. Se quiere conocer la probabilidad que tiene el equipo A de ganar la competición si se sabe que A necesita *i* partidos para ganar y B necesita *j*. Obsérvese que al menos *i*>0 ó *j*>0 para que la situación tenga sentido. Podemos denotar el problema como *prob*(*i*,*j*). Una solución directa es la siguiente:

```
public static float prob (int i, int j) {
   if (i==0)
      return (float)1.0;
   else if (j==0)
      return (float)0.0;
   else
      return (float)((prob(i,j-1)+prob(i-1,j))/2.0);
}
```

Se pretende eliminar la redundancia existente en este algoritmo recursivo múltiple. Durante el resto de la sesión se pide desarrollar un árbol de recursión y su grafo de dependencia asociado, que sean representativos del algoritmo. Puede servirse de SRec o estudiar el código de forma estática.

Entrega

Ambas representaciones gráficas se entregarán al profesor de la asignatura en una hoja (con los apellidos del alumno).

Posteriormente, estas representaciones se usarán como base para aplicar las técnicas de memorización y de tabulación. El plazo de entrega de la práctica completa es el 20 de noviembre de 2007. Debe enviarse por correo electrónico, adjuntando un breve informe siguiendo el modelo disponible en el sitio *Web* de la asignatura. El mensaje debe enviarse a Antonio Pérez Carrasco (antonio.perez.carrasco@urjc.es) con copia a Ángel Velázquez (angel.velazquez@urjc.es).

5. **Cuestionario.** Duración aproximada: 20 minutos.

El alumno debe responder a las preguntas del cuestionario que se entregará en papel.

Cuestionario de opinión sobre el sistema SRec

Nombre y apellidos (opcional): _____

En las preguntas siguientes, marca un valor en cada pregunta. Debes usar un valor de la escala mostrada en la siguiente tabla. Según la clase de pregunta, su significado se referirá a opinión o calidad.

Valor	Opinión	Calidad
1	Nada de acuerdo	Muy mala
2	Poco de acuerdo	Mala
3	Sin opinión	Regular
4	Algo de acuerdo	Buena
5	Totalmente de acuerdo	Muy buena

[ ] Si te parece que SRec es fácil de usar

Las partes que te parecen más difíciles de usar (si las hay) son:

_____

Si te parece que SRec te ha ayudado a analizar los algoritmos recursivos para:

[ ] Determinar las llamada recursivas que se realizan en tiempo de ejecución
[ ] Determinar las dependencias entre llamadas recursivas

[ ] Si te parece alta la calidad en general de SRec para analizar la recursividad

Si te parece alta la calidad de varios aspectos de SRec:

[ ] Iconos
[ ] Controles de animación
[ ] Vista de árbol de activación
[ ] Visor de árboles grandes
[ ] Configuración de formatos
[ ] Configuración de zoom
[ ] Interacción con los paneles (*scroll*, mover paneles, mostrar/ocultar paneles)
[ ] Proceso de generación de una animación
[ ] Proceso de almacenar/cargar una animación
[ ] Visualización almacenada en un fichero de captura

[ ] Si en conjunto te ha gustado SRec

Responde a las siguientes preguntas en formato libre:

1. Di qué características te parece que podrían ser útiles pero SRec carece de ellas:
   _____
   _____
   _____

2. Di qué características de SRec te parecen tan poco útiles que las suprimirías:
   _____
   _____
   _____

3. Describe los aspectos positivos que encuentras en SRec (sobre todo si no se han mencionado antes)
   _____
   _____
   _____

4. Describe los aspectos negativos que encuentras en SRec (sobre todo si no se han mencionado antes)
   _____
   _____
   _____

A2.4 Cuarta Evaluación de SRec

Ingeniería Informática
Asignatura *Diseño y Análisis de Algoritmos*

Curso 2009/2010
Práctica nº 2

Objetivo

El objetivo de la práctica es que el alumno practique la eliminación de la recursividad lineal.

Carácter

La sesión es voluntaria. Puede realizarse individualmente o en parejas, salvo el cuestionario que se realizará individualmente.

Prerrequisitos

El alumno debe tener nociones básicas de eliminación de la recursividad lineal y de análisis de complejidad.

Sesión de laboratorio

SRec es una aplicación para la visualización y animación de métodos recursivos que se encuentra en el sitio *Web* de la asignatura.

La práctica se desarrollará en dos partes, la primera en el laboratorio y la segunda, fuera del mismo. La sesión de laboratorio también sirve para evaluar la calidad de SRec. La sesión consta de 5 fases a realizar secuencialmente:

1. **Demostración del profesor.** Duración aproximada: 10 minutos. Algoritmo: serie recursiva de Fibonacci.

```
public static int fib (int n) {
  if (n==0 || n==1)
     return 1;
  else
     return fib(n-1) + fib(n-2);
}
```

El profesor realizará una demostración del funcionamiento de SRec en la que mostrará, mediante este algoritmo recursivo múltiple, las funciones principales de SRec: seleccionar una máquina virtual de Java, procesar un fichero, generar y reproducir una animación, manejar paneles, almacenar y cargar la animación, y cambiar la configuración.

2. **Primera animación.** Duración aproximada: 15 minutos. Algoritmo: parte entera de un logaritmo.

```
public static int logEntero (int b, int n) {
      if (n<b)
         return 0;
      else
         return 1+logEntero(b,n/b);
   }
```

El alumno generará una animación de este algoritmo recursivo lineal. Para ello, debe realizar las siguientes tareas:

- Procesar la clase *ClasePractica3* (disponible en el sitio *Web* de la asignatura junto con el enunciado de esta práctica y el modelo de informe).
- Generar una animación.
- Experimentar con todos los controles de animación hasta que se comprenda su funcionamiento.
- Almacenar la animación y cargarla de nuevo.

3. **Segunda animación.** Duración aproximada: 25 minutos. Algoritmo: números combinatorios.

```
public static int comb (int m, int n) {
   if (n==0)
      return 1;
   else if (m==n)
      return 1;
   else
      return comb(m-1,n) + comb(m-1,n-1);
}
```

El alumno experimentará con una animación de este algoritmo recursivo múltiple. Esta vez se le pide que realice las siguientes tareas:

- Generar una animación.
- Experimentar con los controles de animación, observando el efecto de cada control sobre cada una de las vistas de la recursividad:
    1. Traza.
    2. Pila de control.
    3. Árbol de activación. (Esta vista, especialmente diseñada para manejar visualizaciones grandes, consta de dos partes: vista global y vista parcial detallada.)
- Experimentar con las opciones de configuración de:

1. Control de la cantidad de información mostrada en cada nodo del árbol.
2. Control de la visualización de la historia pasada.
3. Control del formato gráfico de las 3 visualizaciones (traza, pila de control, árbol de activación).
4. Control del *zoom*.

- Experimentar con las funciones de almacenar:
  1. Una captura.
  2. Una secuencia de capturas (Los ficheros generados pueden contemplarse con el visor de imágenes de Windows).

4. **Tercera animación.** Duración aproximada: 45 minutos. Algoritmo: anillos de un número entero.

Se define el número de anillos de un dígito decimal como el número de trazos cerrados (sean círculos, triángulos, etc.) de su representación escrita. Por ejemplo, el número de anillos de 3 es 0, de 6 es 1 y de 8 es 2. Asimismo, el número de anillos de un entero es igual a la suma del número de anillos de los dígitos de su representación decimal. Se desea diseñar una función *anillos* que, dado un entero, determine cuántos anillos tiene. Por ejemplo:

```
anillos(45)
↓
1
```

correspondiente al único anillo del 4.

Se pide *desarrollar un algoritmo recursivo* que resuelva este problema. Su cabecera será:

```
public static int anillos (int n)
```

Tras la sesión, hay que transformar el algoritmo diseñado en otro equivalente iterativo usando las técnicas explicadas en la teoría.

Entrega

Al final de la sesión se enviará por correo electrónico a Ángel Velázquez (angel.velazquez@urjc.es) un fichero con el nombre y apellidos de los alumnos más lo siguiente:

- Código del algoritmo recursivo.
- Animación generada con SRec que ilustre el comportamiento del algoritmo.

Posteriormente, el alumno debe completar y entregar el informe de la práctica (siguiendo el modelo disponible en el sitio *Web* de la asignatura). Conviene guardar copia

del fichero desarrollado durante la sesión de laboratorio, ya que puede servir para completar el informe de la práctica.

El plazo de entrega del informe es el miércoles 11 de noviembre de 2009, incluido. Debe enviarse por correo electrónico a Ángel Velázquez.

5. **Cuestionario.** Duración aproximada: 15 minutos.
   El alumno debe responder a las preguntas del cuestionario que se entregará en papel.

Evaluación de la práctica

Se evaluará la calidad del algoritmo, la animación, el análisis de complejidad, el informe y, en su caso, la presentación oral en el aula.

Cuestionario de opinión sobre el sistema SRec – I

Nombre y apellidos (opcional): _____

En las preguntas siguientes, marca un valor en cada pregunta. Debes usar un valor de la escala mostrada en la siguiente tabla. Según la clase de pregunta, su significado se referirá a opinión o calidad:

Valor	Opinión	Calidad
1	Nada de acuerdo	Muy mala
2	Poco de acuerdo	Mala
3	Sin opinión	Regular
4	Algo de acuerdo	Buena
5	Totalmente de acuerdo	Muy buena

[ ] Si te parece que SRec es **fácil de usar**
[ ] Si te parece que SRec **te ha ayudado** a ilustrar el comportamiento de algoritmos recursivos
[ ] Si te parece que SRec **incluye las funciones adecuadas** para ilustrar el comportamiento de algoritmos recursivos
[ ] Si en conjunto **te ha gustado** SRec

Responde a las siguientes preguntas en formato libre:

1. Di qué características te parece que podrían ser **útiles** pero SRec carece de ellas:
   _____
   _____
   _____

2. Describe los **aspectos positivos** que encuentras en SRec (sobre todo si no se han mencionado antes):

   _____

   _____

   _____

3. Describe los **aspectos negativos** que encuentras en SRec (sobre todo si no se han mencionado antes)

   _____

   _____

   _____

Ingeniería Informática
Asignatura *Diseño y Análisis de Algoritmos*

Curso 2009/2010
Práctica nº 3

Objetivo

El objetivo de la práctica es que el alumno practique la técnica de divide y vencerás.

Carácter

La sesión es voluntaria. Puede realizarse individualmente o en parejas, salvo el cuestionario que se realizará individualmente.

Prerrequisitos

El alumno debe tener nociones básicas de recursividad, análisis de complejidad y la propia técnica de divide y vencerás.

Enunciado

Sea un vector *v* de números naturales. Se quiere hallar el conjunto de dígitos comunes a todos los números de *v*. Por ejemplo, dado el vector {2348, 1349, 7523, 3215}, la solución es 3.

Para representar el conjunto de dígitos del resultado, podemos suponer que se utiliza un vector de valores booleanos y de tamaño 10. Cada posición contendrá el valor cierto o falso según que el dígito correspondiente sea o no común. Así, la solución del ejemplo anterior se representará con el vector:

{*false*, *false*, *false*, *true*, *false*, *false*, *false*, *false*, *false*, *false*}

La práctica se desarrollará en dos partes, una en el laboratorio y otra, fuera del mismo.

Durante la sesión, se pide a los alumnos:

1. Diseñar un algoritmo de divide y vencerás que resuelva el problema. Su cabecera será:
   ```
   public static boolean[] digitosComunes (int[] v)
   ```
2. Documentar el algoritmo de divide y vencerás.

3. Diseñar un algoritmo iterativo que resuelva el problema.
4. Calcular la complejidad en tiempo de ambos algoritmos y compararlos.

Entrega

Al final de la sesión se enviará por correo electrónico a Ángel Velázquez (angel.velazquez@urjc.es) un fichero con el nombre y apellidos de los alumnos más lo siguiente:

- Código del algoritmo de divide y vencerás.
- Animación generada con SRec que ilustre la definición inductiva del algoritmo.
- Animación generada con SRec que ilustre el comportamiento detallado del algoritmo.

Posteriormente, el alumno debe completar, documentar y entregar el informe de la práctica (siguiendo el modelo disponible en el sitio *Web* de la asignatura). Conviene guardar copia del fichero desarrollado durante la sesión de laboratorio, ya que puede servir para completar el informe de la práctica.

El plazo de entrega del informe es el miércoles 9 de diciembre de 2009, incluido. Debe enviarse por correo electrónico a Ángel Velázquez.

Evaluación

Se evaluará la calidad de los algoritmos, las animaciones y los análisis de complejidad desarrollados, así como la claridad de la memoria.

Cuestionario de opinión sobre el sistema SRec – II

Nombre y apellidos (opcional): _____

En las preguntas siguientes, marca un valor en cada pregunta. Debes usar un valor de la escala mostrada en la siguiente tabla. Según la clase de pregunta, su significado se referirá a opinión o calidad:

Valor	Opinión	Calidad
1	Nada de acuerdo	Muy mala
2	Poco de acuerdo	Mala
3	Sin opinión	Regular
4	Algo de acuerdo	Buena
5	Totalmente de acuerdo	Muy buena

[ ] Si te parece que SRec es **fácil de usar**

Las partes que te parecen **más difíciles de usar** (si las hay) son:

Si te parece que SRec **te ha ayudado**, para algoritmos de divide y vencerás, a:

[ ] Ilustrar la definición inductiva del algoritmo
[ ] Ilustrar el comportamiento detallado del algoritmo

[ ] Si en el tiempo transcurrido desde la práctica 3 (en que también se usó SRec), lo **has usado** para estudiar o resolver **otros problemas de algoritmos recursivos**

En caso afirmativo, explica para qué:

[ ] Si **la calidad general** de SRec te parece alta para ilustrar algoritmos de divide y vencerás

Opina si la **facilidad de uso** y **la calidad** de **distintas partes** de SRec te parecen altas:

	Facilidad de uso	Calidad
Estructura del menú principal		
Iconos		
Vista de árbol de activación		
Visor de árboles grandes		
Vista cronológica		
Vista de estructura de datos		
Control del zoom		
Control de la cantidad de información a mostrar		
Configuración de formatos gráficos		
Controles de animación		
Interacción con los paneles		
Proceso de generación de una animación		
Proceso de almacenar/cargar una visualización/animación		
Visualización/animación almacenada en ficheros		

[ ] Si, en conjunto, **te ha gustado** SRec

Responde a las siguientes preguntas en formato libre:

1. Di qué características te parece que podrían ser **útiles** pero SRec carece de ellas:

   _____

   _____

   _____

2. Di qué características de SRec te parecen tan **poco útiles** que las suprimirías:

   _____

   _____

   _____

A2.5 Quinta Evaluación de SRec

Ingeniería Informática
Asignatura Diseño y Análisis de Algoritmos

Curso 2010/2011
Sesión de introducción a SRec

Objetivo

El objetivo de la sesión es que el alumno se familiarice con el sistema SRec y experimente la diferencia entre algoritmos recursivos redundantes y no redundantes.

Carácter

La sesión es voluntaria. Puede realizarse individualmente o en parejas.

Prerrequisitos

El alumno debe tener nociones básicas de recursividad y la propia técnica de divide y vencerás.

Realización:

SRec es una aplicación para la visualización y animación de métodos recursivos. Se encuentra disponible en el apartado “Contenidos” del Campus Virtual de la Universidad.

La sesión consta de varias fases que se deben realizar secuencialmente. Aparecen descritas a continuación.

1. Primera animación - demostración del profesor. Algoritmo: serie de Fibonacci recursiva.

```
public static int fib (int n) {
   if (n==0 || n==1)
      return 1;
   else
      return fib(n-1) + fib(n-2);
}
```

El profesor realizará una demostración del funcionamiento de SRec en la que mostrará, mediante este algoritmo recursivo múltiple, el funcionamiento de sus funciones principales: seleccionar una máquina virtual de Java, procesar un fichero,

generar y reproducir una animación, manejar paneles, almacenar y cargar la animación, y cambiar las diversas opciones de configuración.

2. Segunda animación. Algoritmo: exponenciación mediante partición binaria del exponente.

```
public static int pot (int b, int e) {
   if (e==0)
      return 1;
   else if (e%2==0)
      return pot(b*b,e/2);
   else
      return b*pot(b*b,e/2);
}
```

El alumno generará una animación de este algoritmo recursivo lineal. Para ello, debe realizar las siguientes tareas:

a) Procesar la clase ClaseEvaluacion disponible en el sitio Web de la asignatura (menú de Archivo > Cargar y procesar clase).

b) Generar una animación (menú de Archivo > Nueva animación...).

c) Experimentar con todos los controles de animación (parte superior derecha de la ventana) hasta que se comprenda su funcionamiento.

d) Almacenar la animación y cargarla de nuevo (en menú de Archivo > Guardar animación... y Archivo > Cargar animación...).

3. Tercera animación. Algoritmo: números combinatorios.

```
public static int comb (int m, int n) {
   if (n==0)
      return 1;
   else if (m==n)
      return 1;
   else
      return comb(m-1,n) + comb(m-1,n-1);
}
```

El alumno experimentará con una animación de este algoritmo recursivo múltiple. Esta vez se le pide que realice las siguientes tareas:

a) Generar una animación.

b) Experimentar con los controles de animación (parte superior derecha de la ventana), observando el efecto de cada operación sobre cada una de las vistas de la recursividad:

   o Traza.

- Pila de control.
- Árbol de activación.

c) Experimentar con las opciones de configuración (menú Visualización [todas las opciones]) de:

- Control de la cantidad de información mostrada en cada nodo del árbol.
- Control de la visualización de la historia pasada.
- Control del formato gráfico de las 3 visualizaciones (traza, pila de control, árbol de activación).

d) Aprender a encontrar la redundancia en los algoritmos recursivos mediante la búsqueda de nodos (y subárboles) iguales (menú Información > Búsqueda de llamadas o botón derecho sobre la subllamada deseada).

4. Cuarta animación. Algoritmo: máximo elemento de un vector de enteros.

```
1     public static int max1 (int[] v)
2     {
3                 return maxDyV1 (v, 0, v.length-1);
4     }
5
6     public static int maxDyV1 (int[] v, int inf, int sup)
7     {
8           if (inf+1>=sup)
9                 return v[inf];
10          else
11          {
12                int medio = (inf+sup)/2;
13
14                int max1 = maxDyV1 (v, inf, medio);
15                int max2 = maxDyV1 (v, medio, sup);
16                return max(max1, max2);
17          }
18    }
19
20    private static int max (int m, int n)
21    {
22          return ( m>n ? m : n );
23    }
```

El alumno debe analizar depurar este programa que implementa el algoritmo de ordenación por mezcla. Puede servirse de SRec o estudiar el código de forma estática.

Debe entregarse un informe a Antonio Pérez Carrasco a través de correo electrónico (antonio.perez.carrasco@urjc.es), en el que especifique los números de línea que contienen errores, cada fragmento de código erróneo y cada nuevo fragmento que sustituye a los fragmentos erróneos.

El informe también incluirá una representación gráfica (hecha manualmente o con ayuda de SRec y de cualquiera de sus vistas) donde se deje ver el mal funcionamiento del algoritmo, acompañada de alguna anotación que explique cuál es el error que aparece en tal representación.

El plazo de entrega es la hora de finalización del laboratorio.

Ingeniería Informática
Asignatura Diseño y Análisis de Algoritmos

Curso 2010/2011
Práctica nº 3

Objetivo

El objetivo de la práctica es que el alumno practique el uso de la técnica de divide y vencerás.

Carácter

La sesión es voluntaria. Puede realizarse individualmente o en parejas.

Prerrequisitos

El alumno debe tener nociones básicas de recursividad y la propia técnica de divide y vencerás.

Enunciado

Sea la siguiente implementación del algoritmo de ordenación por mezcla. Se advierte de que la implementación contiene errores.

```
private static void ordenarPorMezcla (int[] v, int inf, int sup)
{
      if (inf<sup) {
         int medio = (inf+sup)/2;
         ordenarPorMezcla (v, inf, medio-1);
         ordenarPorMezcla (v, medio+1, sup);
         mezclar (v, inf, medio, sup);
      }
}
public  static  void  mezclar  (int[]  v,  int  inf,  int  medio,
                                       int sup){
      int[] vAux = new int[sup-inf+1];      //vector auxiliar
      int i1 = inf;
      int i2 = medio+1;
      int j = 0;  //inicialización de índice del vector auxiliar
      while (i1<=medio && i2<=sup) {
         if (v[i1]>v[i2]) {
            vAux[j] = v[i1];
            i1++;
         }
         else {
            vAux[j] = v[i2];
            i2++;
         }
         j++;
      }
```

```
      for (int i=i1; i<=medio; i++) {
         vAux[j] = v[i];
         j++;
      }
      for (int i=i2; i<=sup; i++) {
         vAux[j] = v[i];
         j++;
      }
      for (int i=inf; i<=sup; i++)
         v[i] = vAux[i-inf]; //transformación de índices y copia
   }
```

El alumno debe depurar este programa que implementa el algoritmo de ordenación por mezcla. Puede servirse de SRec o estudiar el código de forma estática.

Entrega

El alumno debe completar, documentar y entregar el informe de la práctica (siguiendo el modelo disponible en el sitio Web de la asignatura). Conviene guardar copia del fichero desarrollado durante la sesión de laboratorio, ya que sirve para preparar el informe de la práctica.

Junto con el informe se enviará adjunto en el mismo correo electrónico el archivo generado por SRec cuyo nombre es: “log-SREC.txt” (se encuentra en la misma carpeta donde está descomprimida la aplicación, aparece en la primera ejecución de SRec). Este archivo registra la actividad con la herramienta, como por ejemplo los parámetros introducidos para las visualizaciones generadas. El archivo será generado automáticamente durante la sesión del laboratorio, pero deberá actualizarse con el uso que se haga de SRec fuera del laboratorio para finalizar la práctica.

El plazo de entrega del informe y el archivo citado es el jueves 14 de octubre de 2010, incluido. Debe enviarse por correo electrónico a Antonio Pérez Carrasco (antonio.perez.carrasco@urjc.es), el asunto debe ser “DAA: entrega de práctica 3”.

Evaluación

Se evaluará que se hayan encontrado, documentado (con texto e imágenes) y corregido correctamente los errores del programa, así como la claridad de la memoria.

Ingeniería Informática
Asignatura Diseño y Análisis de Algoritmos

Curso 2010/2011
Práctica nº 4

Objetivo

El objetivo de la práctica es que el alumno practique el uso de la técnica de divide y vencerás.

Carácter

La sesión es voluntaria. Puede realizarse individualmente o en parejas, salvo el cuestionario que se realizará individualmente.

Prerrequisitos

El alumno debe tener nociones básicas de recursividad, análisis de complejidad y la propia técnica de divide y vencerás.

Enunciado

Sea un vector v de números naturales. Se quiere hallar el conjunto de dígitos comunes a todos los números de v. Por ejemplo, dado el vector {2348, 1349, 7523, 3215}, la solución es 3, al ser el único dígito que se encuentra en todos los números de dicho vector.

Para representar el conjunto de dígitos del resultado, podemos suponer que se utiliza un vector de valores booleanos y de tamaño 10. Las posiciones, numeradas del 0 al 9, contendrán el valor cierto o falso basándose en si el dígito correspondiente a esa posición es o no común en los números del vector dado inicialmente. Así, la solución del ejemplo anterior se representará con el vector:

```
{false, false, false, true, false, false, false, false, false, false}
 [0]     [1]    [2]    [3]   [4]    [5]    [6]    [7]    [8]    [9]
```

La práctica se desarrollará en dos partes, una en el laboratorio y otra, fuera del mismo. Durante la sesión del laboratorio se pide:

1. Diseñar un algoritmo de divide y vencerás que resuelva el problema. Su cabecera será:

```
public static boolean[] digitosComunes (int[] v)
```

2. Documentar el algoritmo de divide y vencerás, como se indica en la página siguiente.
3. Rellenar un cuestionario sobre SRec disponible en la Web, disponible en la dirección http://www.lite.etsii.urjc.es/cuestionarios/?codigo=srec2010 .

Posteriormente, fuera del laboratorio, los alumnos de cada grupo deberán:

4. Diseñar un algoritmo iterativo que resuelva el problema.
5. Calcular la complejidad en tiempo de ambos algoritmos y compararlos (debe aportarse todo el desarrollo de los cálculos, no únicamente la solución).

Entrega

Al final de la sesión del laboratorio se enviará por correo electrónico a Antonio Pérez Carrasco (antonio.perez.carrasco@urjc.es) un fichero (formato Word) con el nombre y apellidos de los alumnos más lo siguiente:

- Código del algoritmo de divide y vencerás.
- Visualización generada con SRec que ilustre la definición recursiva del algoritmo (formato GIF, PNG o JPG).
- Visualización generada con SRec que ilustre el comportamiento detallado del algoritmo (formato GIF, PNG o JPG).

Junto con el fichero se enviará adjunto en el mismo correo electrónico el archivo generado por SRec cuyo nombre es: “log-SREC.txt” (se encuentra en la misma carpeta donde está descomprimida la aplicación).

Posteriormente, el alumno debe completar, documentar y entregar el informe de la práctica (siguiendo el modelo disponible en el sitio web de la asignatura). Conviene guardar copia del fichero Word desarrollado durante la sesión de laboratorio, ya que sirve para preparar el informe de la práctica.

El plazo de entrega del informe y del archivo citado es el jueves 21 de octubre de 2010, incluido. Debe enviarse igualmente por correo electrónico a Antonio Pérez Carrasco (antonio.perez.carrasco@urjc.es).

Evaluación

Se evaluará la calidad de los algoritmos, las visualizaciones y los análisis de complejidad desarrollados, así como la claridad de la memoria.

Cuestionario de opinión sobre el sistema SRec – II

Nombre y apellidos (opcional): _____

Este es un cuestionario para evaluar la calidad y usabilidad de una herramienta software, SRec. **Esta aplicación ha sido desarrollada por la Universidad con el fin de ayudar a los alumnos en su proceso de aprendizaje de algoritmos**. Para ello, ha contado con la ayuda de tus compañeros de cursos anteriores, que dieron una opinión sincera sobre las cuestiones que se les plantearon. Hoy te pedimos lo mismo a ti, que nos aportes **tu opinión sobre la aplicación SRec para poder seguir mejorándola** y que ello beneficie a tus compañeros del año que viene. Recuerda que la nota de **la práctica NO depende en ningún caso del contenido de tus respuestas en este cuestionario**.

En las preguntas siguientes, marca un valor en cada pregunta. Debes usar un valor de la escala mostrada en la siguiente tabla. Según la clase de pregunta, su significado se referirá a opinión o calidad:

Valor	Opinión	Calidad
1	Nada de acuerdo	Muy mala
2	Poco de acuerdo	Mala
3	Sin opinión	Regular
4	Algo de acuerdo	Buena
5	Totalmente de acuerdo	Muy buena

[ ] Si te parece que SRec es **fácil de usar**

Las partes que te parecen **más difíciles de usar** (si las hay) son:

Si te parece que SRec **te ha ayudado**, para algoritmos de divide y vencerás, a:

[ ] Ilustrar la definición inductiva del algoritmo

[ ] Ilustrar el comportamiento detallado del algoritmo

[ ] Si lo has usado para estudiar o resolver por tu cuenta problemas de algoritmos recursivos

En caso afirmativo, explica para qué:

[ ] Si **la calidad general** de SRec te parece alta para ilustrar algoritmos de divide y vencerás

Opina si la **facilidad de uso** y **la calidad** de **distintas partes** de SRec te parecen altas:

	Facilidad de uso	Calidad
Estructura del menú principal		
Iconos		
Vista de árbol de activación		
Visor de árboles grandes		
Vista cronológica		
Vista de estructura de datos		
Control del zoom		
Control de la cantidad de información a mostrar		
Configuración de formatos gráficos		
Controles de animación		
Interacción con los paneles		
Proceso de generación de una animación		
Proceso de almacenar/cargar una visualización/animación		
Visualización/animación almacenada en ficheros		

[ ] Si, en conjunto, **te ha gustado** SRec

Responde a las siguientes preguntas en formato libre:

1. Di qué características te parece que podrían ser **útiles** pero SRec carece de ellas:
   _____

   _____

   _____

2. Di qué características de SRec te parecen tan **poco útiles** que suprimirías:
   _____

   _____

   _____

Bibliografía

[1] Allen, M. (1995): Estructuras de datos y algoritmos. *Addison-Wesley Iberoamericana,* Capítulo 10, Epígrafe 2.

[2] Alsuwaiyel, M. H. (1999): Algorithms, design techniques and analysis. *World Scientific,* Capítulo 6.

[3] Ben-Bassat, R., Ben-Ari, M. (2007): We work so hard and they don't use it: acceptance of software tools by teachers. *ITiCSE '07: Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, pp. 246-250, New York, NY, USA. ACM.

[4] Ben-Bassat, R., Ben-Ari, M., Uronen, P.A. (2003). The Jeliot 2000 program animation system. *Computers & Education,* 40(1), pp.1-15.

[5] Berghammer, R., Milanese, U. (2001). KIEL: A computer system for visualizing the execution of functional programs. *Proceedings of the International Workshop on Functional and (Constraint) Logic Programming 2001*, Technical Report No. 2017, pp. 365-368, Kiel, Germany: University of Kiel.

[6] Brassard, G., Bratley, P. (1996): Algorítmica, concepción y análisis. *Masson,* Capítulo 4.

[7] Brassard, G., Bratley, P. (1997): Fundamentos de algoritmia. *Prentice Hall,* Capítulo 7.

[8] Chi, E. (2000): A taxonomy of visualization techniques using the data state reference model. *Proceedings IEEE Symposium on Information Visualization 2000 (INFOVIS 2000), pp. 69-75*.

[9] Cormen, T.H., Leiserson, C.E., Rivest, R.L. (2005): Introduction to algorithms. *The MIT Press*.

[10] Dale, N.B., Weems, C. (1991): Pascal, 3rd ed. Lexington, MA: D.C. *Heath*.

[11] De Giusti, A. (2001): Algoritmos, datos y programas con aplicaciones en Pascal, Delphi y Visual Da Vinci. *Prentice Hall,* Capítulo 7.

[12] Dershem, H.L., Erin Parker, D., & Weinhold, R. (1999): A Java function visualizer. *Journal of Computing in Small Colleges*, Vol. 15.

[13] Di Batista, P., Eades, G., Tamassia, T., Toillis, I. (1999): Graph Drawing: Algorithms for the Visualization of Graphs. *Prentice-Hall*, Capítulo 3.

[14] Diehl, S.: Software Visualization (2007): Visualizing the Structure, Behaviour, and Evolution of Software. *Springer*.

[15] Eskola, J., & Tarhio, J. (2002): On a visualization of recursion with Excel. In M. Ben-Ari (Ed.), *Proceedings of the Second Program Visualization Workshop 2002 (PVW 2002), Technical Report DAIMI PB – 567*, pp. 45-51.

[16] Fernández, L., Pérez, A., Velázquez, J. Á., Urquiza, J. (2007): A framework for the automatic generation of algorithm animations based on design techniques. *Proceedings Sixth European Conference on Technology Enhanced Learning (EC-TEL 2007)*, pp.475-480. *Lecture Notes in Computer Science*, Vol. 4753/2007, pp. 475-480.

[17] Fernández, L., Pérez, A., Velázquez, J.Á., Urquiza, J. (2007): Herramienta de generación automática de visualizaciones de técnicas de diseño de algoritmos. *Proceedings I Seminario de Investigación en Tecnologías de la Información aplicadas a la Educación (SITIAE 2007).*

[18] Fernández, L., Pérez, A., Velázquez, J.Á., Urquiza, J. (2007): SRec, un sistema para la animación de árboles de recursión. *Proceedings VIII Simposio Nacional de Tecnologías de la Información y las Comunicaciones en la Educación (SINTICE 2007).*

[19] George, C.E. (2000). EROSI: Visualising recursion and discovering new errors. *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education, SIGCSE'00*, pp. 305-309, New York, ACM Press.

[20] Gonzalo A. (1997): Esquemas algorítmicos: enfoque metodológico y problemas resueltos. *Universidad Nacional de Educación a Distancia* (Capítulo 3).

[21] Götschi, T., Sanders, I., & Galpin, V. (2003): Mental models of recursion. *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, SIGCSE'03*, pp. 346-350, New York, ACM Press.

[22] Götschi, T., Galpin, V., & Sanders, I. (2006): Mental models of recursion revisited. *Proceedings of the 11th Annual Conference on Innovation and*

Technology in Computer Science Education, ITiCSE'06, pp. 138-142, New York, ACM Press.

[23] Harsh Jain. (2006): Java2XML: A Java To XML Converter. Dirección: https://java2xml.dev.java.net (último acceso: 30-mayo-2008; dejó de estar disponible a lo largo de 2009).

[24] Haynes, S.M. (1995): Explaining recursion to the unsophisticated. *ACM SIGCSE Bulletin*, Vol. 27(3), pp. 3-6 y 14.

[25] Hornbæk, K., Bederson, B., Plaisant, C. (2002): Navigation patterns and usability of zoomable user interfaces with and without an overview. *Journal ACM Transactions on Computer-Human Interaction (TOCHI),* Vol. 9(4).

[26] Hundhausen, C. D., Douglas, S. A., and Stasko, J. T. (2002): A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, Vol. 13(3), pp. 259-290.

[27] Ihantola, P., Karavirta, V., Korhonen, A., Nikander, J. (2005): Taxonomy of effortless creation of algorithm visualization. *Proceedings 2005 International Workshop on Computing Education Research*, pp. 123-133, ACM Press.

[28] Infovis.net (revista), http://www.infovis.net/printMag.php?num=100&lang=1 (último acceso, 06 de junio de 2011).

[29] Jadud, M.C. (2006): Methods and tools for exploring novice compilation behaviour. *Proceedings of the second international workshop on Computing education research (ICER 2006)*, pp. 73-84.

[30] Jadud, M.C., Henriksen, P. (2009): Flexible, reusable tools for studying novice programmers. *Proceedings of the fifth international workshop on Computing education research workshop (ICER 2009)*, pp. 37-42.

[31] Jehng, J.-C.J., Tung, S.-H.S. & Chang, C.-T. (1999): A visualization approach to learning the concept of recursion. *Journal of Computer Assisted Learning*, Vol. 15, pp. 279-290.

[32] Johnsonbaugh, R., Schaefer, M. (2004): Algorithms (internacional edition). *Pearson Education*, Capítulo 5.

[33] Kahney, K. (1983): What do novice programmers know about recursion? *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI'83*, pp. 235-239, New York, ACM Press.

[34] Kehoe, C., Stasko, J., Taylor, A. (2001): Rethinking the evaluation of algorithm animations as learning aids: an observational study. *International Journal of Human-Computer Studies*, Vol. 54(2), pp. 265–284.

[35] Kleinberg, J., Tardos, É. (2006): Algorithm design. *Pearson Addison-Wesley*, Capítulo 5.

[36] Kölling, M. (2001): BlueJ extensions. Dirección: http://www.bluej.org/extensions/extensions.html (último acceso: 14-jul-2011).

[37] Kölling, M., Quig, B., Patterson, A., Rosenberg, J. (2003): The BlueJ system and its pedagogy. *Journal of Computer Science Education*, Vol. 13(4), pp. 249-268.

[38] Korhonen, A., Malmi, L. (2000): Algorithm simulation with automatic assessment. *Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE conference on Innovation and technology in computer science education (ITiCSE '00)*. ACM SIGCSE Bulletin, Volume 32 Issue 3, Sept.

[39] Lawrence, A., Badre, A., Stasko, J. T. (1994): Empirically evaluating the use of animations to teach algorithms. *Proceedings of the 1994 IEEE Symposium on Visual Languages*, St. Louis, MO, pp. 48-54.

[40] Lee, R.C.T., Tseng, S.S., Chang R.C., Tsai, Y.T. (2007): Introducción al diseño y análisis de algoritmos, un enfoque estratégico. *Mc Graw Hill*, Capítulo 4.

[41] Levy, D., & Lapidot, T. (2000): Recursively speaking: analyzing students' discourse of recursive phenomena. *Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education, SIGCSE'00*, pp. 315-319, New York, ACM Press.

[42] Levy, D., & Lapidot, T. (2002): Shared terminology, private syntax: The case of recursive descriptions. *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, SIGCSE'02*, pp. 89-93, New York, ACM Press.

[43] Lintern, R., Michaud, J., Storey, M-A., Wu, X. (2003): Plugging-in visualization: experiences integrating a visualization tool with Eclipse. *Proceedings of the 2003 ACM symposium on Software visualization (SOFTVIS 2003)*, pp. 47-56.

[44] Mirolo, C. (2010): Learning (through) recursion: A multidimensional analysis of the competences achieved by CS1 students. *Proceedings of the 15th Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE'10*, pp. 160-164, New York, ACM Press.

[45] Moreno, A., Myller, N., Ben-Ari, M., Sutinen, E. (2004): Program animation in Jeliot 3. *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE 2004), Vol. 36(3).*

[46] Moreno, A., Myller, N., Sutinen, E., and Ben-Ari, M. (2004): Visualizing programs with Jeliot 3. *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI 2004)*, pp. 373-376, Gallipoli (Lecce).

[47] Myller, N., Bednarik, R., Moreno, A. (2007): Integrating dynamic program visualization into BlueJ: the Jeliot 3 Extension. *Proceedings of Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007)*, pp. 505-506.

[48] Naps, T., Cooper, S., Koldehofe, B., Leska, C., Rößling, G., Dann, W., Korhonen, A., Malmi, L., Rantakokko, J., Ross, R., Anderson, J., Fleischer, R., Kuittinen, M., McNally, M. (2003): Evaluating the educational impact of visualization. *ACM SIGCSE Bulletin*, Vol. 35(4), pp. 124-136.

[49] Naps, T., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., Velázquez, J. Á. (2003): Exploring the Role of Visualization and Engagement in Computer Science Education. *ACM SIGCSE Bulletin*, Vol. 35(2), pp. 131-152.

[50] Nielsen, J. (1995): Usability 101: Introduction to Usability. Dirección: http://www.useit.com/alertbox/20030825.html (último acceso: 21-jun-2011)

[51] Nielsen, J. (1993): Usability Engineering. *Morgan Kaufmann*.

[52] Pareja, C., Urquiza, J., Velázquez, J. Á. (2007): WinHIPE: an IDE for functional programming based on rewriting and visualization. *ACM SIGPLAN Notices*, Vol. 42(3), pp. 14–23.

[53] Paterson, J.H., Haddow, J., Nairn, M. (2006): A design patterns extension for the BlueJ IDE. *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education (ITICSE 2006)*, pp. 280-284.

[54] Pérez, A. (2008): Visualización de algoritmos implementados bajo la técnica de Divide y Vencerás. *Proceedings II Seminario de Investigación en Tecnologías de la Información aplicadas a la Educación (SITIAE 2008).*

[55] Pérez, A. (2008): SRec versión 1.0: Manual de uso. *Serie de Informes Técnicos DLSII-URJC*, Vol. 2008-03.

[56] Pérez, A. (2009): SRec versión 1.1: Manual de uso. *Serie de Informes Técnicos DLSII-URJC*, Vol. 2009-02.

[57] Pérez, A. (2010): Fundamentos de usabilidad aplicados a un software de fines docentes. *IV Seminario de Investigación en Tecnologías de la Información aplicadas a la Educación (SITIAE 2010)*, pp. 179-194.

[58] Pérez, A. (2011): SRec como plug-in de visualización de algoritmos para el entorno de programación BlueJ. *V Seminario de Investigación en Tecnologías de la Información aplicadas a la Educación (SITIAE 2011)*, en imprenta.

[59] Pérez, A. (2011): SRec versión 1.2: Manual de uso. *Serie de Informes Técnicos DLSII-URJC*, Vol. 2011-05.

[60] Pérez, A. (2009): SRec, página Web. Dirección: http://www.lite.etsii.urjc.es/srec (último acceso, 4-junio-2011).

[61] Pérez, A., Velázquez, J. Á. (2009): Resultado de las tres primeras evaluaciones de usabilidad de SRec. *Serie de Informes Técnicos DLSI1-URJC*, Vol. 2009-06.

[62] Pérez, A., Velázquez, J. Á. (2009): Resultado de las tres primeras evaluaciones de usabilidad de SRec. *III Seminario de Investigación en Tecnologías de la Información Aplicadas a la Educación (SITIAE 2009), pp. 143-154*.

[63] Pérez, A., Velázquez, J. Á. (2009): Animación automatizada de técnicas de diseño de algoritmos. *XV Jornadas de Enseñanza Universitaria de la Informática (JENUI 2009).*

[64] Pérez, A., Velázquez, J. Á. (2009): SRec, software de animación de la recursividad. *XV Jornadas de Enseñanza Universitaria de la Informática (JENUI 2009),* recurso docente.

[65] Pérez, A., Velázquez, J. Á. (2010): Resultado de la cuarta evaluación de usabilidad de SRec. *Serie de Informes Técnicos DLSI1-URJC,* Vol. 2010-02.

[66] Pérez, A., Velázquez, J. Á. (2010): Entornos para el aprendizaje visual de la programación. *I Jornadas en Innovación y TIC Educativas (JITICE 2010).*

[67] Pérez, A., Velázquez, J. Á. (2011): Resultado de la quinta evaluación de usabilidad de SRec. *Serie de Informes Técnicos DLSI1-URJC,* Vol. 2011-02.

[68] Pérez, A., Velázquez, J. Á. (2011): SRec as a plug-in for Visualizing Recursion. In Proceedings *VI Program Visualization Workshop (PVW 2011).*

[69] Pérez, A., Velázquez, J. Á., Almeida, F. (2011): La representación de algoritmos diseñados bajo la técnica divide y vencerás. *Actas XIII Simposio Internacional de Informática Educativa (SIIE 2011).*

[70] Pérez, A., Velázquez, J. Á., Urquiza, J. (2010): Multiple Usability Evaluations of a Program Animation Tool. *Proceedings of International Conference on Advanced Learning Technologies (ICALT 2010),* pp. 452-454.

[71] Sahni, S. (2005): Data structures, Algorithms and Applications in Java. *Silicon Press,* Capítulo 19.

[72] Saraiya, P., Shaffer, C., McCrickard, S., North, C. (2004): Effective features of algorithm visualizations. In Proceedings of the *35th SIGCSE technical symposium on Computer science education, SIGCSE '04*, pp 382–386, New York, NY, USA, ACM.

[73] Stasko, J. (1997): Using student-built algorithm animations as learning aids. 28th *ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE 97),* San Jose, California, pp. 25-29.

[74] Stasko, J. T., Badre, A., Lewis, C. (1993): Do algorithm animations assist learning? An empirical study and analysis. *Proceedings of ACM INTERCHI 1993 conference of human factors in computing systems*, pp. 61–66.

[75] Stern, L., & Naish, L. (2002): Visual representations for recursive algorithms. *Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education, SIGCSE'02*, pp. 196-200, New York, ACM Press.

[76] Rößling, G., Freisleben, B. (2002): Animal: A system for supporting multiple roles in algorithm animation. *Journal of Visual Languages & Computing*, Vol. 13(3), pp. 341-354.

[77] Rößling, G., Schroeder, P.: Animalipse (2008): An Eclipse plug-in for AnimalScript. *Proceedings of Fifth Program Visualization Workshop (PVW 2008)*, pp. 97-104.

[78] Terada, M. (2005): ETV: a program trace player for students. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education, ITiCSE'05*, pp. 118-122, New York, ACM Press.

[79] Tudoreanu, M. E. (2003): Designing effective program visualization tools for reducing user's cognitive effort. *Proceedings of the 2003 ACM symposium on Software visualization*, pp. 105-114, ACM Press, New York, USA.

[80] Tung, S-H., Chang, C-T., Wong, W-K., Jehng, J-C. (2001): Visual representations for recursion. *International Journal of Human-Computer Studies*, Vol. 54(3), pp. 285-300.

[81] Urquiza, J., Velázquez, J. Á. (2009): A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems. *ACM Transactions on Computing Education (TOCE)*, Vol. 9(2), ACM Press: New York, NY, Article 9.

[82] Velázquez, J Á., Pérez, A. (2009): Aumentando la Interacción con la Visualización de Algoritmos Recursivos. *Actas de X Congreso Internacional de Interacción Persona-Ordenador 2009*.

[83] Velázquez, J. Á., Pérez, A. (2008): Generación y Mantenimiento de Animaciones de Programas con XML. *Actas X Simposio Internacional de Informática Educativa (SIIE 2008)*.

[84] Velázquez, J. Á., Pérez, A. (2009): SRec 1.2: visualizador integrado de programas recursivos generales y de Divide y Vencerás. *Proceedings XI International Symposium on Computers in Education* (SIIE 2009).

[85] Velázquez, J. Á., Pérez, A. (2010): InfoVis Interaction Techniques in Animation of Recursive Programs. *Algorithms* (ISSN 1999-4893), MDPI.

[86] Velázquez, J. Á., Pérez, A. (2011): Interactive learning of recursión by means of animation. Libro "Educational Stages and Interactive Learning: from Kindergarden to Workplace Training", *IGI-Global*. Ed.: Dr. Jiyou Jia.

[87] Velázquez, J. Á., Pérez, A., Debdi, O. (2011): Experiences in Usability Evaluation of Educational Programming Tools. Libro "Student Usability in Educational Software and Games: Improving Experiences", *IGI-Global*. Ed.: Carina S. González.

[88] Velázquez, J. Á., Pérez, A., Lázaro, C., Urquiza, J. (2007): Un sistema con múltiples vistas para la visualización y animación de la recursividad. *Proceedings IX International Symposium on Computers in Education (SIIE 2007)*.

[89] Velázquez, J. Á., Pérez, A., Urquiza, J. (2008): A design of automatic visualizations for divide-and-conquer algorithms. *Electronic Notes in Theoretical Computer Science* (2009), Vol. 224. *Proceedings V Program Visualization Workshop (PVW 2008)*.

[90] Velázquez, J. Á., Pérez, A., Urquiza, J. (2009): Interactive visualization of Recursion with SRec. *14th ACM-SIGCSE Annual Conference on Innovation and Technology in Computer Science (ITICSE 2009)*, póster.

[91] Velázquez, J. Á., Pérez, A., Urquiza, J. (2008): SRec: An animation system of recursion for algorithm courses. *13rd ACM-SIGCSE Annual Conference on Innovation and Technology in Computer Science (ITICSE 2008)*, pp.225-229.

[92] Velázquez, J. Á., Pérez, A., Urquiza, J., Almeida, F. (2010): Sobre la interacción en la visualización del software. *XI Congreso Internacional de Interacción Persona-Ordenador (Interacción 2010)*.

[93] Velázquez, J. Á., Urquiza, J., Almeida, F., Pérez, A. (2009): La visualización y animación de programas como tecnología para el aprendizaje de la

programación. *I Encuentro de Intercambio de Experiencias en Innovación Docente* (celebrado en la Universidad Rey Juan Carlos).

[94] Wilcocks, D., & Sanders, I. (1993): Animating recursion as an aid to instruction. *Computers & Education*, Vol. 23, pp. 221-226.

[95] Wirth, N. (1976): Algorithms + Data Structures = Programs. *Englewood Cliffs, NJ: Prentice Hall* (1976).

[96] Wu, C.-C., Dale, N.B., & Bethel, L.J. (1998): Conceptual models and cognitive learning styles in teaching recursion. *Proceedings of the 29th SIGCSE Technical Symposium on Computer Science Education, SIGCSE'98*, pp. 292-296, New York, ACM Press.

[97] Wu, C.-C., Lee, Lin, J.M.-C., & Hsu I.Y-W. (1996): Closed laboratories using SimList and SimRecur. *Computers & Education*, Vol. 28, pp. 55-64.

[98] Yi, J.S., Kang, Y.A., Stasko, J.T., and Jacko, J.A. (2007): Toward a deeper understanding of the role of interaction in information visualization. *IEEE Trans. Visualization and Computer Graphics*, Vol. 13(6), 1,224-1,231.